09-25-00

A

Please type a plus sign (+) inside this box → +

PTO/SB/05 (4/98)
Approved for use through 09/30/2000 OMB 0651-0032
Patent and Trademark Office. U S DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

09/22/00 JC839 U.S. PTO

jc841 U.S. PTO 09/668610 09/22/00

UTILITY PATENT APPLICATION TRANSMITTAL

Only for new nonprovisional applications under 37 CFR 1 53(b))

Attorney Docket No.	042390.P8104X
First Inventor or Application Identifier	Carl M. Ellison
Title	Protecting Software Environment in Isolated Execution
Express Mail Label No.	EL466333336US

APPLICATION ELEMENTS
See MPEP chapter 600 concerning utility patent application contents

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

1. ☒ Fee Transmittal Form
(Submit an original, and a duplicate for fee processing)
2. ☒ Specification [*Total Pages* 31]
(preferred arrangement set forth below)
   - Descriptive title of the Invention
   - Cross References to Related Applications
   - Statement Regarding Fed sponsored R & D
   - Reference to Microfiche Appendix
   - Background of the Invention
   - Brief Summary of the Invention
   - Brief Description of the Drawings (*if filed*)
   - Detailed Description
   - Claim(s)
   - Abstract of the Disclosure
3. ☒ Drawing(s) *(35 U.S.C. 113)* [*Total Sheets* 13]
4. Oath or Declaration [*Total Pages* 12]
   - a. ☒ Newly executed (original copy)
   - b. ☐ Copy from a prior application (37 C.F.R. § 1.63(d)) *(for continuation/divisional with Box 16 completed)*
     - i. ☐ DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR §§ 1.63(d)(2) and 1.33(b).

*NOTE FOR ITEMS 1 & 13: IN ORDER TO BE ENTITLED TO PAY SMALL ENTITY FEES, A SMALL ENTITY STATEMENT IS REQUIRED (37 C.F.R. § 1.27), EXCEPT IF ONE FILED IN A PRIOR APPLICATION IS RELIED UPON (37 C.F.R. § 1.28).

5. ☐ Microfiche Computer Program *(Appendix)*
6. Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all necessary)*
   - a. ☐ Computer Readable Copy
   - b. ☐ Paper Copy (identical to computer copy)
   - c. ☐ Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

7. ☒ Assignment Papers (cover sheet & document(s))
8. ☐ 37 C.F.R. § 3.73(b) Statement *(when there is an assignee)* ☐ Power of Attorney
9. ☐ English Translation Document *(if applicable)*
10. ☐ Information Disclosure Statement (IDS)/PTO - 1449 ☐ Copies of IDS Citations
11. ☐ Preliminary Amendment
12. ☒ Return Receipt Postcard (MPEP 503) *(Should be specifically itemized)*
13. ☐ *Small Entity Statement(s) ☐ Statement filed in prior application, Status still proper and desired
14. ☐ Certified Copy of Priority Document(s) *(if foreign priority is claimed)*
15. ☐ Other: ....

16. If a CONTINUING APPLICATION, *check appropriate box, and supply the requisite information below and in a preliminary amendment:*
☐ Continuation ☐ Divisional ☒ Continuation-in-part (CIP) of prior application No: 09/540,946
*Prior application Information: Examiner*______ *Group/Art Unit:*______
For CONTINUATION or DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts

17. CORRESPONDENCE ADDRESS

☐ *Customer Number of Bar Code Label* *(Insert Customer No. or Attach bar code label here)* or ☒ *Correspondence address below*

Name	BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP				
Address	12400 Wilshire Boulevard, Seventh Floor				
City	Los Angeles	*State*	California	*Zip Code*	90025
Country	U.S.A.	*Telephone*	(714) 557-3800	*Fax*	(714) 557-3347

Name (Print/Type)	Thinh V. Nguyen, Reg. No. 42,034		
Signature	[illegible]	*Date*	09/22/00

Burden Hour Statement This form is estimated to take 0.2 hours to complete Time will vary depending upon the needs of the individual case Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231 DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS SEND TO Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231

09668610 .092200

Please type a plus sign (+) inside this box → +

FEE TRANSMITTAL for FY 2000

Patent fees are subject to annual revision.
Small Entity payments must be supported by a small entity statement, otherwise large entity fees must be paid. See Forms PTO/SB/09-12. See 37 C.F.R §§ 1.27 and 1.28.

TOTAL AMOUNT OF PAYMENT	($)	1,312.00

Complete if Known

Application Number	
Filing Date	September 22, 2000
First Named Inventor	Carl M. Ellison
Examiner Name	
Group/Art Unit	
Attorney Docket No.	042390.P8104X

JC841 U.S. PTO 09/668610 09/22/00

METHOD OF PAYMENT (check one)

1. [✓] The Commissioner is hereby authorized to charge indicated fees to:
   [✓] The Commissioner is hereby authorized to credit any over payments to:

Deposit Account Number: 02-2666
Deposit Account Name: Blakely, Sokoloff, Taylor & Zafman LLP

[✓] Charge Any Additional Fees Required Under 37 CFR §§ 1.16,1.17, 1.18 and 1.20.

2. [✓] Payment Enclosed:
   [✓ Check] ☐ Money Order ☐ Other

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
101	690	201	345	Utility filing fee	$690.00
106	310	206	155	Design filing fee	
107	480	207	240	Plant filing fee	
108	690	208	345	Reissue filing fee	
114	150	214	75	Provisional filing fee	

SUBTOTAL (1) ($) 690.00

2. EXTRA CLAIM FEES

			Extra Claims		Fee from below		Fee Paid
Total Claims	48	- 20 =	28	X	18.00	=	$504.00
Independent Claims	4	- 3 =	1	X	78.00	=	$78.00
Multiple Dependent						=	

**or number previously paid, if greater, For Reissues, see below

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
103	18	203	9	Claims in excess of 20
102	78	202	39	Independent claims in excess of 3
104	260	204	130	Multiple Dependent claim, if not paid
109	78	209	39	**Reissue independent claims over original patent
110	18	210	9	**Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($) 582.00

FEE CALCULATION (continued)

3. ADDITIONAL FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
105	130	205	65	Surcharge - late filing fee or oath	
127	50	227	25	Surcharge - late provisional filing fee or cover sheet.	
139	130	139	130	Non-English specification	
147	2,520	147	2,520	For filing a request for reexamination	
112	920*	112	920*	Requesting publication of SIR prior to Examiner action	
113	1,840*	113	1,840*	Requesting publication of SIR after Examiner action	
115	110	215	55	Extension for response within first month	
116	380	216	190	Extension for response within second month	
117	870	217	435	Extension for response within third month	
118	1,210	218	680	Extension for response within fourth month	
128	1,850	228	925	Extension for response within fifth month	
119	300	219	150	Notice of Appeal	
120	300	220	150	Filing a brief in support of an appeal	
121	260	221	130	Request for oral hearing	
138	1,510	138	1510	Petition to institute a public use proceeding	
140	110	240	55	Petition to revive - unavoidable	
141	1,210	241	605	Petition to revive - unintentional	
142	1,210	242	605	Utility issue fee (or reissue)	
143	430	243	215	Design issue fee	
144	580	244	290	Plant issue fee	
122	130	122	130	Petitions to the Commissioner	
123	50	123	50	Petitions related to provisional applications	
126	240	126	240	Submission of Information Disclosure Stmt	
581	40	581	40	Recording each patent assignment per property (times number of properties)	40.00
146	790	246	395	Filing a submission after final rejection (37 CFR 1.129(a))	
149	790	249	395	For each additional invention to be examined (37 CFR 1.129(b))	
				Other fee (specify) ____________	
				Other fee (specify) ____________	

* Reduced by Basic Filing Fee Paid

SUBTOTAL (3) ($) 40.00

SUBMITTED BY				Complete (if applicable)	
Typed or Printed Name	Thinh V. Nguyen			Reg. Number	42,034
Signature	[illegible]	Date	09/22/00	Deposit Account User ID	02-2666

Docket No. 042390.P8104X
Express Mail No.: EL466331471US

UNITED STATES PATENT APPLICATION

FOR

PROTECTING SOFTWARE ENVIRONMENT IN ISOLATED EXECUTION

INVENTORS:

CARL M. ELLISON
ROGER A. GOLLIVER
HOWARD C. HERBERT
DERRICK C. LIN
FRANCIS X. MCKEEN
GIL NEIGER
KEN RENERIS
JAMES A. SUTTON
SHREEKANT S. THAKKAR
MILLIND MITTAL

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 Wilshire Blvd., 7th Floor
Los Angeles, CA 90025-1026
(714) 557-3800

CROSS-REFERENCES TO RELATED APPLICATIONS

This is a continuation-in-part of U.S. Patent Application No. 09/540,946 filed March 31, 2000.

BACKGROUND

1. FIELD OF THE INVENTION

This invention relates to microprocessors. In particular, the invention relates to processor security.

2. DESCRIPTION OF RELATED ART

Advances in microprocessor and communication technologies have opened up many opportunities for applications that go beyond the traditional ways of doing business. Electronic commerce (E-commerce) and business-to-business (B2B) transactions are now becoming popular, reaching the global markets at a fast rate. Unfortunately, while modern microprocessor systems provide users convenient and efficient methods of doing business, communicating and transacting, they are also vulnerable to unscrupulous attacks. Examples of these attacks include virus, intrusion, security breach, and tampering, to name a few. Computer security, therefore, is becoming more and more important to protect the integrity of the computer systems and increase the trust of users.

Threats caused by unscrupulous attacks may be in a number of forms. Attacks may be remote without requiring physical accesses. An invasive remote-launched attack by hackers may disrupt the normal operation of a system connected to thousands or even millions of users. A virus program may corrupt code and/or data of a single-user platform.

Existing techniques to protect against attacks have a number of drawbacks. Anti-virus programs can only scan and detect known viruses. Most anti-virus programs use a weak policy in which a file or program is assumed good until proved bad. For many security applications, this weak policy may not be appropriate. In addition, most anti-virus programs are used locally where they are resident in the platform. This may not be suitable in a group work environment. Security co-processors or smart cards using cryptographic or other security techniques have limitations in speed performance, memory capacity, and flexibility. Redesigning operating systems creates software compatibility issues and causes tremendous investment in development efforts.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will become apparent from the following detailed description of the present invention in which:

Figure 1A is a diagram illustrating a logical operating architecture according to one embodiment of the invention.

Figure 1B is a diagram illustrating accessibility of various elements in the operating system and the processor according to one embodiment of the invention.

Figure 1C is a diagram illustrating a computer system in which one embodiment of the invention can be practiced.

Figure 2 is a diagram illustrating a secure platform according to one embodiment of the invention.

Figure 3A is a diagram illustrating a subset of a software environment having a usage protector according to one embodiment of the invention.

Figure 3B is a diagram illustrating a subset of a software environment having a usage protector according to another embodiment of the invention.

Figure 3C is a diagram illustrating the subset of a software environment according to yet another embodiment of the invention.

Figure 3D is a diagram illustrating the subset of a software environment according to yet another embodiment of the invention.

Figure 3E is a diagram illustrating the subset of a software environment according to yet another embodiment of the invention.

Figure 4 is a flowchart illustrating a process to protect usage of a subset of a software environment according to one embodiment of the invention.

Figure 5 is a flowchart illustrating the process to protect usage of the subset according to another embodiment of the invention.

Figure 6 is a flowchart illustrating the process to protect usage of the subset according to yet another embodiment of the invention.

Figure 7 is a flowchart illustrating a process to protect usage of the subset according to yet another embodiment of the invention.

DETAILED DESCRIPTION

In the following description, for purposes of explanation, numerous details are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that these specific details are not required in order to practice the present invention. In other instances, well-known electrical structures and circuits are shown in block diagram form in order not to obscure the present invention.

ARCHITECTURE OVERVIEW

One principle for providing security in a computer system or platform is the concept of an isolated execution architecture. The isolated execution architecture includes logical and physical definitions of hardware and software components that interact directly or indirectly with an operating system of the computer system or platform. An operating system and the processor may have several levels of hierarchy, referred to as rings, corresponding to various operational modes. A ring is a logical division of hardware and software components that are designed to perform dedicated tasks within the operating system. The division is typically based on the degree or level of privilege, namely, the ability to make changes to the platform. For example, a ring-0 is the innermost ring, being at the highest level of the hierarchy. Ring-0 encompasses the most critical, privileged components. In addition, modules in Ring-0 can also access to lesser privileged data, but not vice versa. Ring-3 is the outermost ring, being at the lowest level of the hierarchy. Ring-3 typically encompasses users or applications level and has the least privilege. Ring-1 and ring-2 represent the intermediate rings with decreasing levels of privilege.

Figure 1A is a diagram illustrating a logical operating architecture 50 according to one embodiment of the invention. The logical operating architecture 50 is an abstraction of the components of an operating system and the processor. The logical operating architecture 50 includes ring-0 10, ring-1 20, ring-2 30, ring-3 40, and a processor nub loader 52. The processor nub loader 52 is an instance of an processor executive (PE) handler. The PE handler is used to handle and/or manage a processor executive (PE) as will be discussed later. The logical operating architecture 50 has two modes of operation: normal execution mode and isolated execution mode. Each ring in the logical operating architecture 50 can operate in both modes. The processor nub loader 52 operates only in the isolated execution mode.

Ring-0 10 includes two portions: a normal execution Ring-0 11 and an isolated execution Ring-0 15. The normal execution Ring-0 11 includes software modules that are critical for the operating system, usually referred to as kernel. These software modules include primary operating system (e.g., kernel) 12, software drivers 13, and hardware drivers 14. The isolated execution Ring-0 15 includes an operating system (OS) nub 16 and a processor nub 18. The OS nub 16 and the processor nub 18 are instances of an OS executive (OSE) and processor executive (PE), respectively. The OSE and the PE are part of executive entities that operate in a secure environment associated with the isolated area 70 and the isolated execution mode. The processor nub loader 52 is a protected bootstrap loader code held within a chipset in the system and is responsible for loading the processor nub 18 from the processor or chipset into an isolated area as will be explained later.

Similarly, ring-1 20, ring-2 30, and ring-3 40 include normal execution ring-1 21, ring-2 31, ring-3 41, and isolated execution ring-1 25, ring-2 35, and ring-3 45, respectively. In particular, normal execution ring-3 includes N applications 42_1 to 42_N and isolated execution ring-3 includes K applets 46_1 to 46_K.

One concept of the isolated execution architecture is the creation of an isolated region in the system memory, referred to as an isolated area, which is protected by both the processor and chipset in the computer system. The isolated region may also be in cache memory, protected by a translation look aside (TLB) access check. Access to this isolated region is permitted only from a front side bus (FSB) of the processor, using special bus (e.g., memory read and write) cycles, referred to as isolated read and write cycles. The special bus cycles are also used for snooping. The isolated read and write cycles are issued by the processor executing in an isolated execution mode. The isolated execution mode is initialized using a privileged instruction in the processor, combined with the processor nub loader 52. The processor nub loader 52 verifies and loads a ring-0 nub software module (e.g., processor nub 18) into the isolated area. The processor nub 18 provides hardware-related services for the isolated execution.

One task of the processor nub 18 is to verify and load the ring-0 OS nub 16 into the isolated area, and to generate the root of a key hierarchy unique to a combination of the platform, the processor nub 18, and the operating system nub 16. The operating system nub 16 provides links to services in the primary OS 12 (e.g., the unprotected segments of the operating system), provides page management within the isolated area, and has the responsibility for loading ring-3 application modules 45, including applets

46_1 to 46_K, into protected pages allocated in the isolated area. The operating system nub 16 may also load ring-0 supporting modules.

The operating system nub 16 may choose to support paging of data between the isolated area and ordinary (e.g., non-isolated) memory. If so, then the operating system nub 16 is also responsible for encrypting and hashing the isolated area pages before evicting the page to the ordinary memory, and for checking the page contents upon restoration of the page. The isolated mode applets 46_1 to 46_K and their data are tamper-resistant and monitor-resistant from all software attacks from other applets, as well as from non-isolated-space applications (e.g., 42_1 to 42_N), dynamic link libraries (DLLs), drivers and even the primary operating system 12. Only the processor nub 18 or the operating system nub 16 can interfere with or monitor the applet's execution.

Figure 1B is a diagram illustrating accessibility of various elements in the operating system 10 and the processor according to one embodiment of the invention. For illustration purposes, only elements of ring-0 10 and ring-3 40 are shown. The various elements in the logical operating architecture 50 access an accessible physical memory 60 according to their ring hierarchy and the execution mode.

The accessible physical memory 60 includes an isolated area 70 and a non-isolated area 80. The isolated area 70 includes applet pages 72 and nub pages 74. The non-isolated area 80 includes application pages 82 and operating system pages 84. The isolated area 70 is accessible only to elements of the operating system and processor operating in isolated execution mode. The non-isolated area 80 is accessible to all elements of the ring-0 operating system and to the processor.

The normal execution ring-0 11 including the primary OS 12, the software drivers 13, and the hardware drivers 14, can access both the OS pages 84 and the application pages 82. The normal execution ring-3, including applications 42_1 to 42_N, can access only to the application pages 82. Both the normal execution ring-0 11 and ring-3 41, however, cannot access the isolated area 70.

The isolated execution ring-0 15, including the OS nub 16 and the processor nub 18, can access to both of the isolated area 70, including the applet pages 72 and the nub pages 74, and the non-isolated area 80, including the application pages 82 and the OS pages 84. The isolated execution ring-3 45, including applets 46_1 to 46_K, can access only to the application pages 82 and the applet pages 72. The applets 46_1 to 46_K reside in the isolated area 70.

Figure 1C is a diagram illustrating a computer system 100 in which one embodiment of the invention can be practiced. The computer system 100 includes a processor 110, a host bus 120, a memory controller hub (MCH) 130, a system memory 140, an input/output controller hub (ICH) 150, a non-volatile memory, or system flash, 160, a mass storage device 170, input/output devices 175, a token bus 180, a motherboard (MB) token 182, a reader 184, and a token 186. The MCH 130 may be integrated into a chipset that integrates multiple functionalities such as the isolated execution mode, host-to-peripheral bus interface, memory control. Similarly, the ICH 150 may also be integrated into a chipset together or separate from the MCH 130 to perform I/O functions. For clarity, not all the peripheral buses are shown. It is contemplated that the system 100 may also include peripheral buses such as Peripheral Component Interconnect (PCI), accelerated graphics port (AGP), Industry Standard Architecture (ISA) bus, and Universal Serial Bus (USB), etc.

The processor 110 represents a central processing unit of any type of architecture, such as complex instruction set computers (CISC), reduced instruction set computers (RISC), very long instruction word (VLIW), or hybrid architecture. In one embodiment, the processor 110 is compatible with an Intel Architecture (IA) processor, such as the Pentium^{TM} series, the IA-32^{TM} and the IA-64^{TM}. The processor 110 includes a normal execution mode 112 and an isolated execution circuit 115. The normal execution mode 112 is the mode in which the processor 110 operates in a non-secure environment, or a normal environment without the security features provided by the isolated execution mode. The isolated execution circuit 115 provides a mechanism to allow the processor 110 to operate in an isolated execution mode. The isolated execution circuit 115 provides hardware and software support for the isolated execution mode. This support includes configuration for isolated execution, definition of an isolated area, definition (e.g., decoding and execution) of isolated instructions, generation of isolated access bus cycles, and generation of isolated mode interrupts.

In one embodiment, the computer system 100 can be a single processor system, such as a desktop computer, which has only one main central processing unit, e.g. processor 110. In other embodiments, the computer system 100 can include multiple processors, e.g. processors 110, 110a, 110b, etc., as shown in Figure 1C. Thus, the computer system 100 can be a multi-processor computer system having any number of processors. For example, the multi-processor computer system 100 can operate as part of a server or workstation environment. The basic description and operation of processor 110 will be discussed in detail below. It will be appreciated by those skilled in the art that the basic description and operation of processor 110 applies to the other processors 110a and 110b, shown in Figure 1C, as well as any number of other processors that may be utilized in the multi-processor computer system 100 according to one embodiment of the present invention.

The processor 110 may also have multiple logical processors. A logical processor, sometimes referred to as a thread, is a functional unit within a physical processor having an architectural state and physical resources allocated according to some partitioning policy. Within the context of the present invention, the terms "thread" and "logical processor" are used to mean the same thing. A multi-threaded processor is a processor having multiple threads or multiple logical processors. A multi-processor system (e.g., the system comprising the processors 110, 110a, and 110b) may have multiple multi-threaded processors.

The host bus 120 provides interface signals to allow the processor 110 or processors 110, 100a, and 110b to communicate with other processors or devices, e.g., the MCH 130. In addition to normal mode, the host bus 120 provides an isolated access bus mode with corresponding interface signals for memory read and write cycles when the processor 110 is configured in the isolated execution mode. The isolated access bus mode is asserted on memory accesses initiated while the processor 110 is in the isolated execution mode. The isolated access bus mode is also asserted on instruction pre-fetch and cache write-back cycles if the address is within the isolated area address range and the processor 110 is initialized in the isolated execution mode. The processor 110 responds to snoop cycles to a cached address within the isolated area address range if the isolated access bus cycle is asserted and the processor 110 is initialized into the isolated execution mode.

The MCH 130 provides control and configuration of memory and input/output devices such as the system memory 140 and the ICH 150. The MCH 130 provides interface circuits to recognize and service isolated access assertions on memory reference bus cycles, including isolated memory read and write cycles. In addition, the MCH 130 has memory range registers (e.g., base and length registers) to represent the isolated area in the system memory 140. Once configured, the MCH 130 aborts any access to the isolated area that does not have the isolated access bus mode asserted.

The system memory 140 stores system code and data. The system memory 140 is typically implemented with dynamic random access memory (DRAM) or static random access memory (SRAM). The system memory 140 includes the accessible physical memory 60 (shown in Figure 1B). The accessible physical memory includes a loaded operating system 142, the isolated area 70 (shown in Figure 1B), and an isolated control and status space 148. The loaded operating system 142 is the portion of the operating system that is loaded into the system memory 140. The loaded OS 142 is typically loaded from a mass storage device via some boot code in a boot storage such as a boot read only memory (ROM). The isolated area 70, as shown in Figure 1B, is the memory area that is defined by the processor 110 when operating in the isolated execution mode. Access to the isolated area 70 is restricted and is enforced by the processor 110 and/or the MCH 130 or other chipset that integrates the isolated area functionalities. The isolated control and status space 148 is an input/output (I/O)-like, independent address space defined by the processor 110 and/or the MCH 130. The isolated control and status space 148 contains mainly the isolated execution control and status registers. The isolated control and status space 148 does not overlap any existing address space and is accessed using the isolated bus cycles. The system memory 140 may also include other programs or data which are not shown.

The ICH 150 represents a known single point in the system having the isolated execution functionality. For clarity, only one ICH 150 is shown. The system 100 may have many ICH's similar to the ICH 150. When there are multiple ICH's, a designated ICH is selected to control the isolated area configuration and status. In one embodiment, this selection is performed by an external strapping pin. As is known by one skilled in the art, other methods of selecting can be used, including using programmable configuring registers. The ICH 150 has a number of functionalities that are designed to support the isolated execution mode in addition to the traditional I/O functions. In particular, the ICH 150 includes an isolated bus cycle interface 152, the processor nub loader 52 (shown in Figure 1A), a digest memory 154, a cryptographic key storage 155, an isolated execution logical processor manager 156, and a token bus interface 159.

The isolated bus cycle interface 152 includes circuitry to interface to the isolated bus cycle signals to recognize and service isolated bus cycles, such as the isolated read and write bus cycles. The processor nub loader 52, as shown in Figure 1A, includes a processor nub loader code and its digest (e.g., hash) value. The processor nub loader 52 is invoked by execution of an appropriate isolated instruction (e.g., Iso_Init) and is transferred to the isolated area 70. From the isolated area 80, the processor nub loader 52 copies the processor nub 18 from the system flash memory (e.g., the processor nub code 18 in non-volatile memory 160) into the isolated area 70, verifies and logs its integrity, and manages a symmetric key used to protect the processor nub's secrets. In one embodiment, the processor nub loader 52 is implemented in read only memory (ROM). For security purposes, the processor nub loader 52 is unchanging, tamper-resistant and non-substitutable. The digest memory 154, typically implemented in RAM, stores the digest (e.g., hash) values of the loaded processor nub 18, the operating system nub 16, and any other critical modules (e.g., ring-0 modules) loaded into the isolated execution space. The cryptographic key storage 155 holds a symmetric encryption/decryption key that is unique for the platform of the system 100. In one embodiment, the cryptographic key storage 155 includes internal fuses that are programmed at manufacturing. Alternatively, the cryptographic key storage 155 may also be created with a random number generator and a strap of a pin. The isolated execution logical processor manager 156 manages the operation of logical processors operating in isolated execution mode. In one embodiment, the isolated execution logical processor manager 156 includes a logical processor count register that tracks the number of logical processors participating in the isolated execution mode. The token bus interface 159 interfaces to the token bus 180. A combination of the processor nub loader digest, the processor nub digest, the operating system nub digest, and optionally additional digests, represents the overall isolated execution digest, referred to as isolated digest. The isolated digest is a fingerprint identifying the ring-0 code controlling the isolated execution configuration and operation. The isolated digest is used to attest or prove the state of the current isolated execution.

The non-volatile memory 160 stores non-volatile information. Typically, the non-volatile memory 160 is implemented in flash memory. The non-volatile memory 160 includes the processor nub 18. The processor nub 18 provides the initial set-up and low-level management of the isolated area 70 (in the system memory 140), including verification, loading, and logging of the operating system nub 16, and the management of the symmetric key used to protect the operating system nub's secrets. The processor nub 18 may also provide application programming interface (API) abstractions to low-level security services provided by other hardware. The processor nub 18 may also be distributed by the original equipment manufacturer (OEM) or operating system vendor (OSV) via a boot disk.

The mass storage device 170 stores archive information such as code (e.g., processor nub 18), programs, files, data, applications (e.g., applications 42_1 to 42_N), applets (e.g., applets 46_1 to 46_K) and operating systems. The mass storage device 170 may include compact disk (CD) ROM 172, floppy diskettes 174, and hard drive 176, and any other magnetic or optical storage devices. The mass storage device 170 provides a mechanism to read machine-readable media. When implemented in software, the elements of the present invention are the code segments to perform the necessary tasks. The program or code segments can be stored in a processor readable medium or transmitted by a computer data signal embodied in a carrier wave, or a signal modulated by a carrier, over a transmission medium. The "processor readable medium" may include any medium that can store or transfer information. Examples of the processor readable medium include an electronic circuit, a semiconductor memory device, a ROM, a flash memory, an erasable programmable ROM (EPROM), a floppy diskette, a compact disk CD-ROM, an optical disk, a hard disk, a fiber optical medium, a radio frequency (RF) link, etc. The computer data signal may include any signal that can propagate over a transmission medium such as electronic network channels, optical fibers, air, electromagnetic, RF links, etc. The code segments may be downloaded via computer networks such as the Internet, an Intranet, etc.

I/O devices 175 may include any I/O devices to perform I/O functions. Examples of I/O devices 175 include a controller for input devices (e.g., keyboard, mouse, trackball, pointing device), media card (e.g., audio, video, graphics), a network card, and any other peripheral controllers.

The token bus 180 provides an interface between the ICH 150 and various tokens in the system. A token is a device that performs dedicated input/output functions with security functionalities. A token has characteristics similar to a smart card, including at least one reserved-purpose public/private key pair and the ability to sign data with the private key. Examples of tokens connected to the token bus 180 include a motherboard token 182, a token reader 184, and other portable tokens 186 (e.g., smart card). The token bus interface 159 in the ICH 150 connects through the token bus 180 to the ICH 150 and ensures that when commanded to prove the state of the isolated execution, the corresponding token (e.g., the motherboard token 182, the token 186) signs only valid isolated digest information. For purposes of security, the token should be connected to the digest memory.

PROTECTING SOFTWARE ENVIRONMENT IN ISOLATED EXECUTION

The overall architecture discussed above provides a basic insight into a hierarchical executive architecture to manage a secure platform. The elements shown in Figures 1A, 1B, and 1C are instances of an abstract model of this hierarchical executive architecture. The implementation of this hierarchical executive architecture is a combination of hardware and software. In what follows, the processor executive, the processor executive handler, and the operating system executive are abstract models of the processor nub 18, the processor nub loader 52, and the operating system nub 16 (Figures 1A, 1B, and 1C), respectively.

Figure 2 is a diagram illustrating a secure platform 200 according to one embodiment of the invention. The secure platform 200 includes the OS nub 16, the processor nub 18, a key generator 240, a hashing function 220, and a usage protector 250, all operating within the isolated execution environment, as well as a software environment 210 that may exist either inside or outside the isolated execution environment.

The OS nub or OS executive (OSE) 16 is part of the operating system running on the secure platform 200. The OS nub 16 has an associated OS nub identifier (ID) 201, that may be delivered with the OS nub 16 or derived from an OS nub code or associated information. The OS nub ID 201 may be a pre-determined code that identifies the particular version of the OS nub 16. It may also represent a family of various versions of the OS nub 16. The OS nub 16 may optionally have access to a public and private key pair unique for the platform 200. The key pair may be generated and stored at the time of manufacturing, at first system boot, or later. The protected private key 204 may be programmed into the fuses of a cryptographic key storage 155 of the input/output control hub (ICH) 150 or elsewhere in persistent storage within the platform 200. The protected private key 204 may be based upon a random number generated by an external random number generator. In one embodiment, the protected private key 204 is generated by the platform 200 itself the first time the platform 200 is powered up. The platform 200 includes a random number generator to create random numbers. When the platform 200 is first powered up, a random number is generated upon which the protected private key 204 is based. The protected private key 204 can then be stored in the multiple key storage 164 of the non-volatile flash memory160. The feature of the protected private key 204 is that it cannot be calculated from its associated public key 205. Only the OS nub 16 can retrieve and decrypt the encrypted private key for subsequent use. In the digital signature generation process, the protected private key 204 is used as an encryption key to encrypt a digest of a message producing a signature, and the public key 205 is used as a decryption key to decrypt the signature, revealing the digest value.

The processor nub 18 includes a master binding key (BK0) 202. The BK0 202 is generated at random when the processor nub 18 is first invoked, i.e., when it is first executed on the secure platform 200. The key generator 240 generates a key operating system nub key (OSNK) 203 which is provided only to the OS Nub 16. The OS nub 16 may supply the OSNK 203 to trusted agents, such as the usage protector 250. The key generator 240 receives the OS Nub identifier 201 and the BK0 202 to generate the OSNK 203. There are a number of ways for the key generator 240 to generate the OSNK 203. The key generator 240 generates the OSNK 203 by combining the BK0 202 and the OS Nub ID 201 using a cryptographic hash function. In one embodiment, the OS nub ID 201 identifies the OS nub 16 being installed into the secure platform 200. The OS nub ID 201 can be the hash of the OS nub 16, or a hash of a certificate that authenticates the OS nub 16, or an ID value extracted from a certificate that authenticates the OS nub 16. It is noted that a cryptographic hash function is a one-way function, mathematical or otherwise, which takes a variable-length input string, called a pre-image and converts it to a fixed-length, generally smaller, output string referred to as a hash value. The hash function is one-way in that it is difficult to generate a pre-image that matches the hash value of another pre-image. In one embodiment, the OS nub ID 201 is a hash value of one of the OS Nub 16 and a certificate representing the OS nub 16. Since the security of an algorithm rests in the key, it is important to choose a strong cryptographic process when generating a key. The software environment 210 may include a plurality of subsets (e.g., subset 230). The usage of the software environment 210 or the usage of the subset 230 is protected by the usage protector 250. The usage protector 250 uses the OSNK 203 to protect the usage of the subset 230. The software environment 210 may include an operating system (e.g., a Windows operating system, a Windows 95 operating system, a Windows 98 operating system, a Windows NT operating system, Windows 2000 operating system) or a data base. The subset 230 may be a registry in the Windows operating system or a subset of a database. Elements can be implemented in hardware or software.

The subset 230 is hashed by the hashing function 220 to produce a first hash value 206 and a second hash value 312. One way to detect intrusion or modification of the subset 230 is to compare the state of the subset before and after a time period. The first and second hash values 206 and 312 are typically generated at different times and/or at different places.

The usage protector 250 is coupled to the key generator 240 to protect usage of the software environment 210 or the subset 230, using the OSNK 203. The usage protection includes protection against unauthorized reads, and detection of intrusion, tampering or unauthorized modification. If the two hash values are not the same, then the usage protector 250 knows that there is a change in the subset 230. If it is known that this change is authorized and an updated hash value has been provided, the usage protector 250 would merely report the result. Otherwise, the usage protector 250 may generate an error or a fault function. When a user is notified of the error, fault condition, he or she would know that the subset 230 has been tampered, modified. The user may take appropriate action. Therefore, the usage of the subset 230 is protected.

The are several different embodiments of the usage protector 250. In one embodiment, the usage protector 250 decrypts the subset using the OSNK 203. In two other embodiments, the usage protector 250 uses not only the OSNK 203 but also the first hash value 206 and the second hash value 312. Yet in two other embodiments, the usage protector uses the OSNK 203, the protected private key 204 and the pubic key 205.

Figure 3A is a diagram illustrating the usage protector 250 shown in Figure 2 according to one embodiment of the invention. The usage protector 250 includes a compressor 370, an encryptor 375, a storage 380, a decryptor 385, and a compressor 390.

The compressor 370 receives the subset 230 and compresses the subset 230 to generate a compressed subset 372. The encryptor 375 then encrypts the compressed subset 372 using the OSNK 203, producing the encrypted compressed subset 377. The OSNK 203 is provided to the usage protection 250 by the key generator 240 as shown in Figure 2. At a later time, a request can be made to the OS nub 16 to access the encrypted compressed subset 377. If this request is granted, the decryptor 385 decrypts the retrieved encrypted compressed subset 382 using the OSNK 203, producing the retrieved compressed subset 387. The decompressor 390 then expands the retrieved compressed subset 387 to produce the retrieved subset 392. The compression operation is applied to save time (i.e. speed up) and/or space for storing the subset 230 in a memory. In another embodiment, the encryptor 375 takes the subset 230 without going through the compressing process and encrypts the subset 230 to generate an encrypted subset using the OSNK 203, and the decryptor 385 decrypts the retrieved encrypted subset directly producing the retrieved subset 392. The encrypting of the compressed subset 372 or the subset 230 prevents unauthorized reads of the subset 230.

Figure 3B is a diagram illustrating the usage protector 250 shown in Fig. 2 according to another embodiment of the invention. The usage protector 250 includes an encryptor 305, a decryptor 365, a storage medium 310, and a comparator 315.

The encryptor 305 encrypts the first hash value 206 using the OSNK 203 to generate an encrypted first hash value 302. The encrypted first hash value 302 is then stored in the storage 310. Storage medium 310 may be any type of medium capable of storing the encrypted hash value 302. The storage medium 310 may be any type of disk, i.e., floppy disks, hard disks and optical disks) or any type of tape, i.e., tapes. At a later time, the subset 230 is tested for integrity. The decryptor 365 decrypts the retrieved encrypted first hash value 303 using the OSNK 203. This decrypting process generates a decrypted hash value 366. This decrypted first hash value 366 is then compared to the second hash value 312 by the comparator 315 to detect if changes have been made in the subset 230. If the two values match, then subset 230 has not been changed. If the subset 230 is deliberately updated by an authorized agent, the stored encrypted hash value is also updated, and a subsequent integrity test again results in the two hash values (366 and 312) matching. If the subset 230 is modified by an unauthorized agent that does not update the stored encrypted hash value, then the subsequent integrity test results in differing hash values 366 and 312, signaling the unauthorized modification. The unauthorized agent cannot avoid this detection by attempting to generate its own version of the stored encrypted hash value, because the unauthorized agent does not have access to the OSNK 203.

Figure 3C is a diagram illustrating the usage protector 250 shown in Figure 2 according to one embodiment of the invention. The usage protector 250 includes a decryptor 325, a signature generator 320, a storage medium 322, and a signature verifier 330.

The decryptor 325 accepts the OS nub's encrypted private key (i.e., protected) 204, and decrypts it using the OSNK 203, exposing the private key 328 for use in the isolated environment. The signature generator 320 generates a signature 304 for the subset 230 using the private key 328. It is noted that the subset 230 may be compressed before input it into the signature generator 320 to generate the signature 304. The signature algorithm used by the signature generator 320 may be public-key digital signature algorithm which makes use of a secret private key to generate the signature, and a public key to verify the signature. Example algorithms include ElGama, Schnorr and Digital Signature Algorithms schemes just to name a few. In one embodiment, the generation of the signature 304 includes hashing the subset 230 to generate a before hash value, which is then encrypted using the private key 328 to generate the signature 304. The signature 304 is then saved in a storage medium 322. At a later time, the signature is retrieved from the storage 322, and the retrieved signature 306 is used, along with public key 205, by the signature verifier 330 to verify the subset 230. The signature verifier 330 verifies whether the subset 230 has been modified, producing a modified/not-modified indicator 331. In one embodiment, the verification process includes decrypting the retrieved signature 306 using the public key 205 to expose the before hash value. The subset 230 is hashed to generate an after hash value. The before hash value is compared to the after hash value to detect whether the subset 230 has been modified. If the two hash values match, the subset 230 is the same as it was when the signature was generated.

Figure 3D is a diagram illustrating the usage protector 250 according to yet another embodiment of the invention. The usage protector 250 includes a decryptor 345, a manifest generator 335, a signature generator 340, a storage medium 349, a signature verifier 350, and a manifest verifier 355.

The decryptor 345 accepts the OS nub's encrypted protected private key 204, and decrypts it using the OSNK 203, exposing the private key 348 for use in the isolated environment. The manifest generator 335 generates a manifest 307 for the subset 230. The manifest 307 represents the subset 230 in a concise manner. The manifest 307 may include a number of descriptors or entities, which characterize some relevant aspects of the subset 230. Typically, these relevant aspects are particular or specific to the subset 230 so that two different subsets have different manifests. In one embodiment, the manifest 307 represents a plurality of entities (i.e., a collection of entities) where each entry in the manifest 307 represents a hash (e.g., unique fingerprint) over one entity in the collection. The subset 230 is partitioned into one or more group where each group has a pointer and associated hash in the manifest 307. The manifest 307 is stored in a storage medium 349 for later use. The manifest 307 is also input to the signature generator 340 to generate a signature 308 over the manifest 307 using the private key 348. The generated signature 308 is also stored in a storage medium 349. At a later time, we desire to verify that the portions of the subset 230 described by the manifest have not changed. This requires verifying that the manifest itself has not been changed, and that each group in subset 230 described by the manifest has not been changed. The stored signature and manifest are retrieved from the storage medium 349. The retrieved signature 309, and the retrieved manifest 354, along with the public key 205, are used by the signature verifier 350 to test that the retrieved manifest 354 is unchanged from the original manifest 307. The signature verifier 350 produces a signature-verified flag 351, which is asserted only if the signature verifies that the manifest is unchanged. In one embodiment, the verification process includes decrypting the retrieved signature 309 using the public key 205 to expose the before hash value. The retrieved manifest 354 is hashed to generate an after hash value. The before hash value is compared to the after hash value to detect whether the retrieved manifest 354 has been modified. If the two hash values match, the retrieved manifest 354 is the same as it was when the signature was generated. The retrieved manifest 354 is also supplied to the manifest verifier 355, which uses the descriptive information in the retrieved manifest 354 to selectively verify portions of subset 230. In a typical embodiment, this involves hashing each group in subset 230, where the group is identified by information in the retrieved manifest 354, and comparing the newly generated hash value against the hash value for the group stored in the retrieved manifest 354. The manifest verifier 355 produces a manifest-verified flag 356, which is asserted if all entries described by the retrieved manifest 354 are verified as unchanged. If both the manifest-verified flag and the signature-verified flag are asserted, then the overall verification process passes, and the selected portions of subset 230 are known to be the same as when the signed manifest was originally generated, and a pass/fail flag 358 is asserted. Note that the signature verifier 350 and the manifest verifier 355 can be invoked in any order.

Figure 3E is a diagram illustrating the usage protector 250 shown in Figure 2 according to another embodiment of the invention. The usage protector 250 includes a first encryptor 305, a second encryptor 365, a storage medium 310, and a comparator 315.

The first encryptor 305 encrypts the first hash value 206 using the OSNK 203. The first hash value 206 is provided by the hashing function 220 as shown in Figure 2. The first encryptor 305 takes the first hash value 206 and encrypts it to generate an encrypted first hash value 302 using the OSNK 203. The encrypted hash value 302 is then stored in a storage 310 for later use. The encryption by the OSNK 203 allows the encrypted first hash value 302 to be stored in arbitrary (i.e., unprotected) storage media. Storage medium 310 may be any type of medium capable of storing information (e.g., the encrypted hash value 302). The storage medium 310 may be any type of disk, i.e., floppy disks, hard disks and optical disks) or any type of tape, i.e., tapes. The second encryptor 365 encrypts the second hash value 312 to generate an encrypted second hash value 301 using the OSNK 203. The second hash value 312 is provided by the hash function 220. The first encryptor 305 and the second encryptor 365 use the same encryption algorithm, and this algorithm produces identical repeatable results for a given input. The encrypted first hash value 302 is now retrieved from the storage 310 for comparing with the encrypted second hash value 301. The comparator 315 compares the encrypted second hash value 301 with the retrieved encrypted first hash value 303 to detect if the subset 230 has been modified or tampered with. In the case where the subset 230 is deliberately updated by an authorized agent, the stored encrypted hash value is also updated. Since the modification of the subset 230 is authorized, the second encrypted hash value 301 is the same as the retrieved first encrypted first hash value 303. In the case where the subset 230 has been unauthorized modified or tampered with, the comparator 315 generates a modified/not-modified flag indicating the subset 230 has been modified and therefore, the subset 230 should not be used. An attacker cannot simply replace the first encrypted hash value 303 with one corresponding to the unauthorized modified subset 230, because the attacker does not have access to the OSNK 203 used to encrypt the hash value.

Figure 4 is a flowchart illustrating a process 400 to protect usage of the software environment 210 or subset 230 according to one embodiment of the invention.

Upon START, the process 400 checks to see whether accessing to the subset is authorized (Block 405). If accessing to the subset is authorized, the process 400 is terminated. Otherwise, the process 400 checks to see if the accessing to the subset is a read or a write (Block 410). If it is a write, the process 400 obtains the OSNK and the subset (Block 415). The process 400 then encrypts the subset using the OSNK (Block 420) and stores the encrypted subset in a storage (Block 425), the process 400 is terminated. If accessing to the subset is a read, the process 400 obtains the OSNK and the encrypted subset (Block 430). The process then decrypts the encrypted subset using the OSNK (Block 435), the process 400 is the terminated.

Figure 5 is a flowchart illustrating a process 500 to protect usage of the subset of the software environment according to one embodiment of the invention.

Upon START, the process 500 checks to see whether the subset is to be updated or to be tested (Block 505). If it is to be updated, the process 500 checks to see whether accessing the subset is authorized (Block 510). If accessing the subset is not authorized, the process 500 is terminated. If accessing the subset is authorized, the process 500 obtains the OSNK and the first hash value (Block 515) and encrypts the first hash value using the OSNK (Block 520). The process 500 then stores the encrypted first hash value in a storage for later use (Block 525). The process is terminated. If the subset is to be tested, the process 500 then obtains the OSNK and the second hash value (Block 550). The process 500 also retrieves the encrypted first hash value from the storage (Block 555) and decrypts it using the OSNK (Block 560). The process 500 then checks to see whether the two hash values are equal (Block 565). If the two hash values are equal, the process 500 clears "modified" flag (Block 570) and is terminated. Otherwise, the process 500 sets "modified" flag (Block 575) and is terminated.

Figure 6 is a flowchart illustrating a process to protect usage of a subset of a software environment according to yet another embodiment of the invention.

Upon START, the process 600 checks to see whether the subset is to be updated or to be tested (Block 605). If it is to be updated, the process 600 checks to see whether the request to be updated is authorized (Block 610). If it is not authorized, the process 600 is terminated. Otherwise, the process 600 obtains the OSNK, the protected private key and the subset (Block 615). The process 600 then decrypts the protected private using the OSNK (Block 620) and signs the subset using the private key (Block 625). The process 600 stores the signature in a storage (Block 630) and is terminated. If the subset is to be tested, the process 600 obtains the public key, the subset, and the signature (Block 650). The process 600 verifies the subset against the signature using the public key (Block 655). The process 600 then checks whether the subset is verified (Block 660). If the subset is verified, the process 600 clears "modified" flag (Block 665) and is terminated. Otherwise, the process 600 sets "modified" flag (Block 670) and is terminated.

Figure 7 is a flowchart illustrating a process 700 to protect usage of a subset of a software environment according to yet another embodiment of the invention.

Upon START, the process 700 checks to see whether the subset is to be updated or tested (Block 705). If it is to be updated, the process 700 then checks to see whether the accessing to the subset is authorized (Block 710). If it is not authorized, the process 700 is terminated. Otherwise, the process 700 obtains the OSNK and the first hash value (Block 715) and encrypts the first hash value using the OSNK (Block 720). The process then stores the encrypted first hash value in a storage (Block 725) and is terminated. If the subset is to be tested, the process 700 obtains the OSNK and the second hash value (Block 750) and encrypts the second hash value using the OSNK (Block 755). The process 700 then retrieves the encrypted first hash value from the storage (Block 760) and checks to see whether the encrypted hashes are equal (Block 765). If they are equal, the process 700 clears "modified" flag and is terminated. Otherwise, the process 700 sets "modified" flag (Block 775) and is terminated.

While this invention has been described with reference to illustrative embodiments, this description is not intended to be construed in a limiting sense. Various modifications of the illustrative embodiments, as well as other embodiments of the invention, which are apparent to persons skilled in the art to which the invention pertains are deemed to lie within the spirit and scope of the invention.

CLAIMS

What is claimed is:

1. An apparatus comprising:

a key generator to generate an operating system nub key (OSNK) unique to an operating system (OS) nub, the OS nub being part of an operating system running on a secure platform; and

a usage protector coupled to the key generator to protect usage of a subset of a software environment using the OSNK.

2. The apparatus of claim 1 wherein the key generator comprises:

a combiner to combine an identification of the OS nub and a master binding key (BK0) of the secure platform, the combined identification and the BK0 corresponding to the OSNK.

3. The apparatus of claim 2 wherein the identification is a hash value of one of the OS nub and a certificate representing the OS nub.

4. The apparatus of claim 1 wherein the usage protector comprises:

an encryptor to encrypt the subset of the software environment using the OSNK, the encrypted subset being stored in a storage; and

a decryptor to decrypt the encrypted subset using the OSNK, the encrypted subset being retrieved from the storage.

5. The apparatus of claim 1 wherein the usage protector comprises:

an encryptor to encrypt a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;

a decryptor to decrypt the encrypted first hash value using the OSNK, the encrypted first hash value being retrieved from the storage; and

a comparator to compare the decrypted first hash value to a second hash value to generate a compared result, the compared result indicating whether the subset of the software environment has been modified.

6. The apparatus of claim 1 wherein the usage protector comprises:
a first encryptor to encrypt a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;
a second encryptor to encrypt a second hash value using the OSNK; and
a comparator to compare the encrypted second hash value to the encrypted first hash value to generate a compared result, the encrypted first hash value being retrieved from the storage, the compared result indicating whether the subset of the software environment has been modified.

7. The apparatus of claim 1 wherein the usage protector comprises:
a decryptor to decrypt a protected private key to generate a private key using the OSNK;
a signature generator coupled to the decryptor to generate a signature of the subset of the software environment using the private key, the signature being stored in a storage; and
a signature verifier to verify the signature to generate a modified/not modified flag using a public key, the signature being retrieved from the storage, the modified/not modified flag indicating whether the subset has been modified.

8. The apparatus of claim 1 wherein the usage protector comprises:
a manifest generator to generate a manifest of the subset of the software environment, the manifest describing the subset of the software environment, the manifest being stored in a storage;
a signature generator coupled to the manifest generator to generate a manifest signature using a private key, the private key being decrypted by a decryptor using the OSNK, the manifest signature being stored in the storage;
a signature verifier to verify the manifest signature to generate a signature verified flag using a public key, the manifest signature being retrieved from the storage; and
a manifest verifier to verify the manifest to generate a manifest verified flag, the manifest being retrieved from the storage, the manifest verified flag and the signature verified flag being tested at a test center, the test center generating a pass/fail signal to indicate whether the subset has been modified.

9. The apparatus of claim 1 wherein the secure platform uses an isolated execution mode.

10. The apparatus of claim 1 wherein the software environment is one of a Windows operating system, a Windows 95 operating system, a Windows 98 operating system, a Windows NT operating system, and a Windows 2000 operating system.

11. The apparatus of claim 1 wherein the subset of the software environment is a registry of an operating system.

12. The apparatus of claim 2 wherein the BK0 is generated at random on a first invocation of a processor nub.

13. A method comprising:

generating an operating system nub key (OSNK) unique to an operating system (OS) nub, the OS nub being part of an operating system running on a secure platform; and

protecting usage of a subset of the software environment using the OSNK.

14. The method of claim 11 wherein generating the OSNK comprises:

combining an identification of the OS nub and a master binding key (BK0) of the secure platform, the combined identification and the BK0 corresponding to the OSNK.

15. The method of claim 14 wherein the identification is a hash value of one of the OS nub and a certificate representing the OS nub.

16. The method of claim 13 wherein protecting usage comprises:

encrypting the subset of the software environment using the OSNK;

storing the encrypted subset in a storage; and

decrypting the encrypted subset from the storage using the OSNK.

17. The method of claim 13 wherein protecting usage comprises:
encrypting a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;
decrypting the encrypted first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being retrieved from the storage; and
comparing the decrypted first hash value to a second hash value to generate a compared result, the decrypted first hash value being retrieved from the storage, the compared result indicating whether the subset of the software environment has been modified.

18. The method of claim 13 wherein protecting usage comprises:
encrypting a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;
encrypting a second hash value using the OSNK; and
comparing the encrypted first hash value to the encrypted second hash value to generate a compared result, the encrypted first hash value being retrieved from the storage, the compared result indicating whether the subset of the software environment has been modified.

19. The method of claim 13 wherein protecting usage comprises:
decrypting a protected private key to generate a private key using the OSNK;
generating a signature of the subset of the software environment using the private key, the signature being stored in a storage; and
verifying the signature to generate a modified/not modified flag using a public key, the signature being retrieved from the storage, the modified/not modified flag indicating whether the subset of the software environment has been modified.

20. The method of claim 13 wherein detecting comprises:
generating a manifest of the subset of the software environment, the manifest describing the subset of the software environment, the manifest being stored in a storage;

generating a manifest signature of the manifest using a private key, the private key being decrypted using the OSNK, the manifest signature being stored in the storage;

verifying the manifest signature to generate a signature verified flag using a public key, the manifest signature being retrieved from the storage; and

verifying the manifest to generate a manifest verified flag, the manifest being retrieved from the storage, the manifest verified flag and the signature verified flag being tested at a test center, the test center generating a pass/fail signal, the pass/fail signal indicating whether the subset of the software environment has been modified.

21. The method of claim 13 wherein the secure platform uses an isolated execution mode.

22. The method of claim 13 wherein the software environment is one of a Windows operating system, a Windows 95 operating system, a Windows 98 operating system, a Windows NT operating system, and a Windows 2000 operating system.

23. The method of claim 13 wherein the subset of the software environment is a registry of the operating system.

24. The method of claim 14, wherein the BK0 is generated at random on a first invocation of a processor nub.

25. A computer program product comprising:

a computer usable medium having computer program code embodied therein, the computer program product having:

computer readable program code for generating an operating system nub key (OSNK) unique to an operating system (OS) nub, the OS nub being part of an operating system running on a secure platform; and

computer readable program code for protecting usage a subset of the software environment using the OSNK.

26. The computer program product of claim 25 wherein the computer readable program code for generating the OSNK comprises:

computer readable program code for combining an identification of the OS nub and a master binding key (BK0) of the secure platform, the combined identification and the BK0 corresponding to the OSNK.

27. The computer program product of claim 26 wherein the identification is a hash value of one of the OS nub and a certificate representing the OS nub.

28. The computer program product of claim 25 wherein the computer readable program code for protecting usage comprises:

computer readable program code for encrypting the subset of the software environment using the OSNK;

computer readable program code for storing the encrypted subset; and

computer readable program code for decrypting the encrypted subset from the storage using the OSNK.

29. The computer program product of claim 25 wherein the computer readable program code for protecting usage comprises:

computer readable program code for encrypting a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;

computer readable program code for decrypting the encrypted first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being retrieved from the storage; and

computer readable program code for comparing the decrypted first hash value to a second hash value to generate a compared result, the decrypted first hash value being retrieved from the storage, the compared result indicating whether the subset of the software environment has been modified.

30. The computer program product of claim 25 wherein the computer readable program code for protecting usage comprises:

computer readable program code for encrypting a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;

computer readable program code for encrypting a second hash value using the OSNK; and

computer readable program code for comparing the encrypted first hash value to the encrypted second hash value to generate a compared result, the encrypted first hash value being retrieved from the storage, the compared result indicating whether the subset of the software environment has been modified.

31. The computer program product of claim 25 wherein the computer readable program code for protecting usage comprises:

computer readable program code for decrypting a protected private key to generate a private key using the OSNK;

computer readable program code for generating a signature of the subset of the software environment using the private key, the signature being stored in a storage; and

computer readable program code for verifying the signature to generate a modified/not modified flag using a public key, the signature being retrieved from the storage, the modified/not modified flag indicating whether the software environment has been modified.

32. The computer program product of claim 25 wherein the computer readable program code for protecting usage comprises:

computer readable program code for generating a manifest of the subset of the software environment, the manifest being stored in a storage;

computer readable program code for generating a manifest signature of the manifest using a private key, the private key being decrypted using the OSNK, the manifest signature being stored in the storage;

computer readable program code for verifying the manifest signature to generate a signature verified flag using a public key, the manifest signature being retrieved from the storage; and

computer readable program code for verifying the manifest to generate a manifest verified flag , the manifest being retrieved from the storage, the manifest verified flag and the signature verified flag being tested at a test center, the test center generating a pass/fail signal, the pass/fail signal indicating whether the subset of the software environment has been modified.

33. The computer program product of claim 25 wherein the secure platform uses an isolated execution mode.

34. The computer program product of claim 25 wherein the software environment is one of a Windows operating system, a Windows 95 operating system, a Windows 98 operating system, a Windows NT operating system, and a Windows 2000 operating system.

35. The computer program product of claim 25 wherein the subset of the software environment is a registry of an operating system.

36. The computer program product of claim 26 wherein the BK0 is generated at random on a first invocation of a processor nub.

37. A system comprising:

a processor;

a storage device coupled to the processor, the storage storing a subset of a software environment; and

a usage protector comprising:

a key generator to generate an operating system nub key (OSNK) unique to an operating system (OS) nub, the operating system nub being part of a software environment running on a secure platform; and

a usage protector coupled to the key generator to protect usage of a subset of the software environment using the OSNK.

38. The system of claim 37 wherein the key generator comprises:

a combiner to combine an identification of the operating system nub and a master binding key (BK0) of the secure platform, the combined identification and BK0 corresponding to the OSNK.

39. The system of claim 38 wherein the identification is a hash value of one of the OS nub and a certificate representing the OS nub.

40. The system of claim 37 wherein the usage protector comprises:
an encryptor to encrypt the subset of the software environment using the OSNK, the encrypted subset being stored in a storage; and
a decryptor to decrypt the encrypted subset using the OSNK, the encrypted subset being retrieved from the storage.

41. The system of claim 37 wherein the usage protector comprises:
an encryptor to encrypt a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;
a decryptor to decrypt the encrypted first hash value using the OSNK, the encrypted first hash value being retrieved from the storage; and
a comparator to compare the decrypted first hash value to a second hash value to generate a compared result, the compared result indicating whether the subset of the software environment has been modified.

42. The system of claim 37 wherein the usage protector comprises:
a first encryptor to encrypt a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;
a second encryptor to encrypt a second hash value using the OSNK; and
a comparator to compare the encrypted second hash value to the encrypted first hash value to generate a compared result, the encrypted first hash value being retrieved from the storage, the compared result indicating whether the subset of the software environment has been modified.

43. The system of claim 37 wherein the usage protector comprises:
a decryptor to decrypt a protected private key to generate a private key using the OSNK;
a signature generator coupled to the decryptor to generate a signature of the subset of the software environment using the private key, the signature being stored in a storage; and
a signature verifier to verify the signature to generate a modified/not modified flag using a public key, the signature being retrieved from the storage, the modified/not

modified flag indicating whether the subset of the software environment has been modified.

44. The system of claim 37 wherein the usage protector comprises:

a manifest generator to generate a manifest of the subset of the software environment, the manifest describing the subset of the software environment, the manifest being stored in a storage;

a signature generator coupled to the manifest generator to generate a manifest signature of the manifest using a private key, the private key being decrypted using the OSNK, the manifest signature being stored in the storage;

a signature verifier to verify the manifest signature to generate a signature verified flag using a public key, the manifest signature being retrieved from the storage; and

a manifest verifier to verify the manifest to generate a manifest verified flag, the manifest being retrieved from the storage, the manifest verified flag and the signature verified flag being tested by a test center, the test center generating a pass/fail signal indicating whether the subset has been modified.

45. The system of claim 37 wherein the secure platform uses an isolated execution mode.

46. The system of claim 37 wherein the software environment is one of a Windows operating system, a Windows 95 operating system, a Windows 98 operating system, a Windows NT operating system, and a Windows 2000 operating system.

47. The system of claim 37 wherein the subset of the software environment is a registry of an operating system.

48. The system of claim 38 wherein the BK0 is generated at random on a first invocation of a processor nub.

ABSTRACT OF THE DISCLOSURE

The present invention is a method and apparatus to protect a subset of a software environment. A key generator generates an operating system nub key (OSNK). The OSNK is unique to an operating system (OS) nub. The OS nub is part of an operating system in a secure platform. A usage protector uses the OSNK to protect usage of a subset of the software environment.

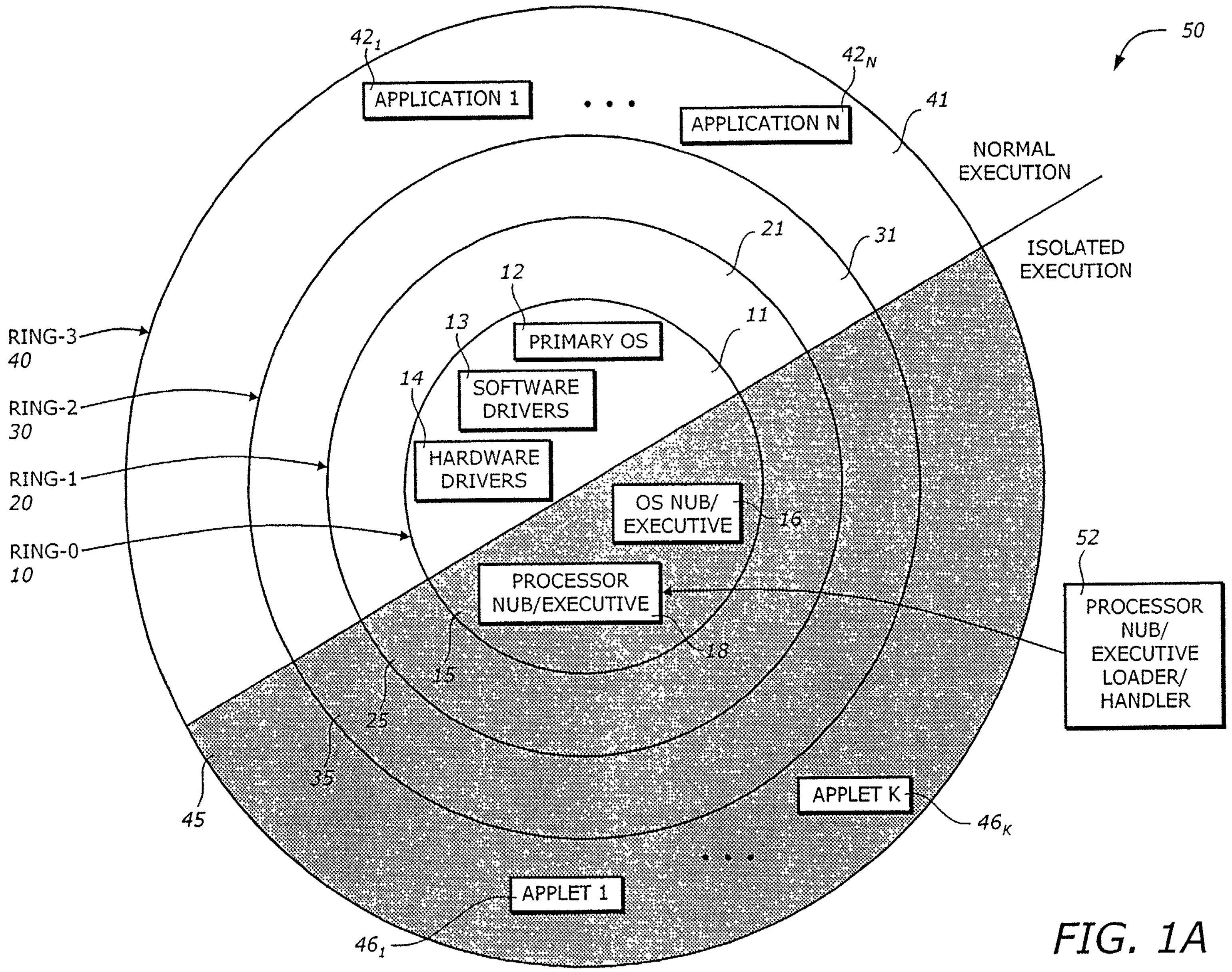

50
42$_1$
APPLICATION 1
. . .
42$_N$
APPLICATION N
41
NORMAL EXECUTION
ISOLATED EXECUTION
21
31
12
PRIMARY OS
13
SOFTWARE DRIVERS
14
HARDWARE DRIVERS
11
RING-3 40
RING-2 30
RING-1 20
RING-0 10
OS NUB/ EXECUTIVE
16
PROCESSOR NUB/EXECUTIVE
18
15
25
35
45
52
PROCESSOR NUB/ EXECUTIVE LOADER/ HANDLER
APPLET K
46$_K$
. . .
APPLET 1
46$_1$

FIG. 1A

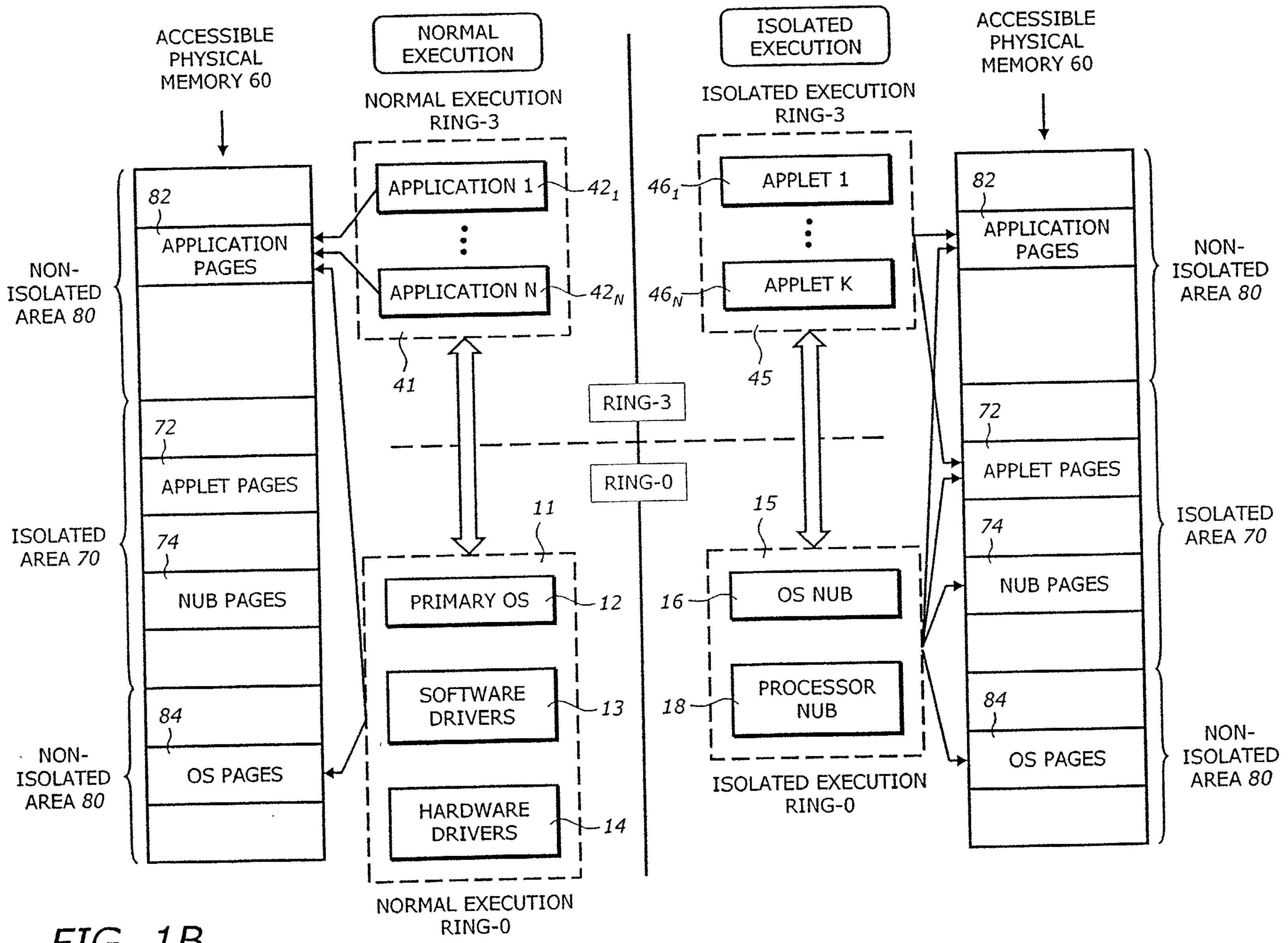

ACCESSIBLE PHYSICAL MEMORY 60
NORMAL EXECUTION
NORMAL EXECUTION RING-3
APPLICATION 1
42₁
APPLICATION N
42N
41
NON-ISOLATED AREA 80
82
APPLICATION PAGES
72
APPLET PAGES
ISOLATED AREA 70
74
NUB PAGES
84
OS PAGES
NON-ISOLATED AREA 80
11
PRIMARY OS
12
SOFTWARE DRIVERS
13
HARDWARE DRIVERS
14
NORMAL EXECUTION RING-0
RING-3
RING-0
ISOLATED EXECUTION
ISOLATED EXECUTION RING-3
46₁
APPLET 1
46N
APPLET K
45
15
16
OS NUB
18
PROCESSOR NUB
ISOLATED EXECUTION RING-0
ACCESSIBLE PHYSICAL MEMORY 60
NON-ISOLATED AREA 80
ISOLATED AREA 70
NON-ISOLATED AREA 80

FIG. 1B

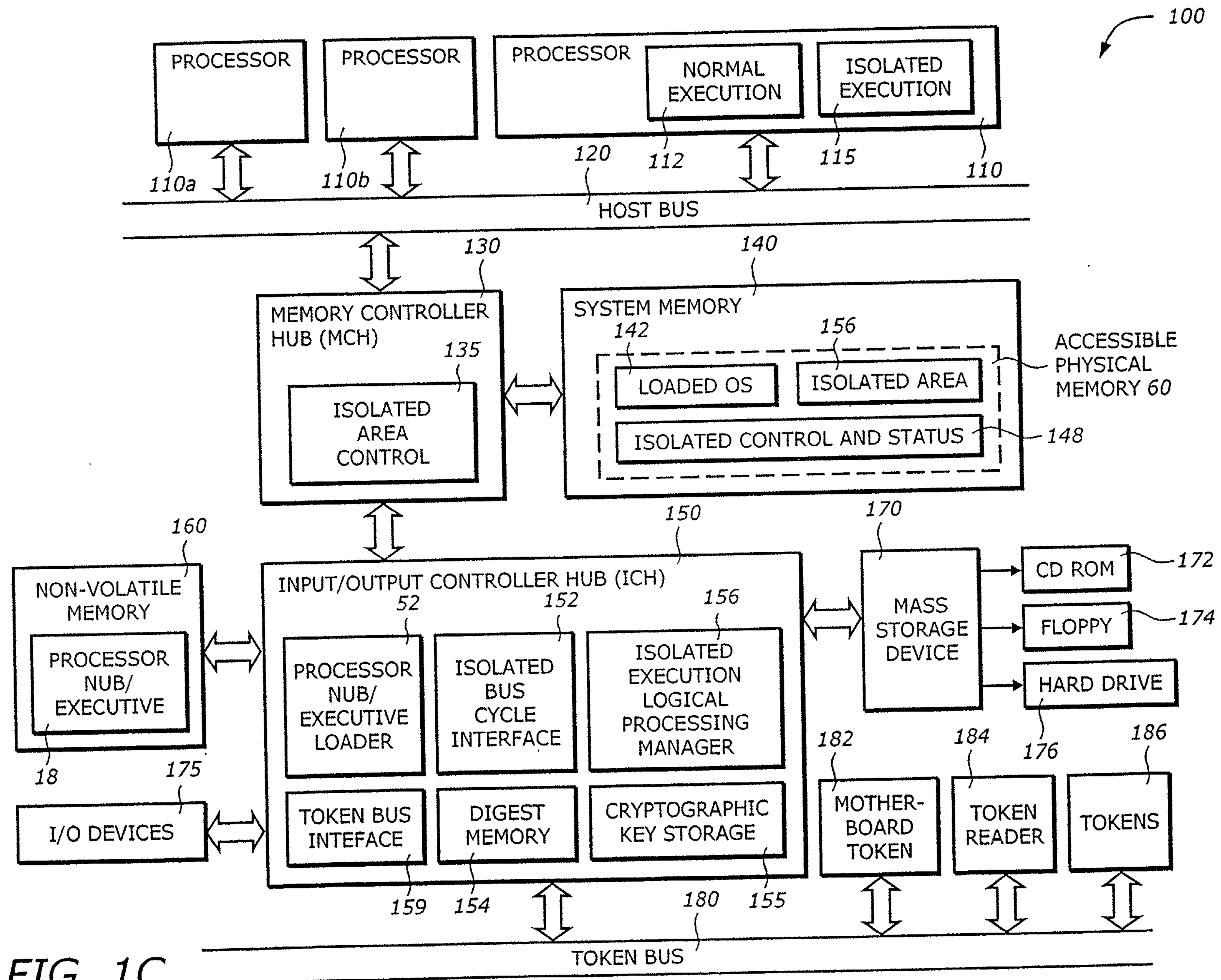

100
PROCESSOR
110a
PROCESSOR
110b
PROCESSOR
NORMAL EXECUTION
112
ISOLATED EXECUTION
115
110
120
HOST BUS
130
MEMORY CONTROLLER HUB (MCH)
135
ISOLATED AREA CONTROL
140
SYSTEM MEMORY
142
LOADED OS
156
ISOLATED AREA
ISOLATED CONTROL AND STATUS
148
ACCESSIBLE PHYSICAL MEMORY 60
160
NON-VOLATILE MEMORY
PROCESSOR NUB/ EXECUTIVE
18
175
I/O DEVICES
150
INPUT/OUTPUT CONTROLLER HUB (ICH)
52
PROCESSOR NUB/ EXECUTIVE LOADER
152
ISOLATED BUS CYCLE INTERFACE
156
ISOLATED EXECUTION LOGICAL PROCESSING MANAGER
TOKEN BUS INTEFACE
159
DIGEST MEMORY
154
CRYPTOGRAPHIC KEY STORAGE
155
170
MASS STORAGE DEVICE
CD ROM
172
FLOPPY
174
HARD DRIVE
176
182
MOTHER-BOARD TOKEN
184
TOKEN READER
186
TOKENS
180
TOKEN BUS

FIG. 1C

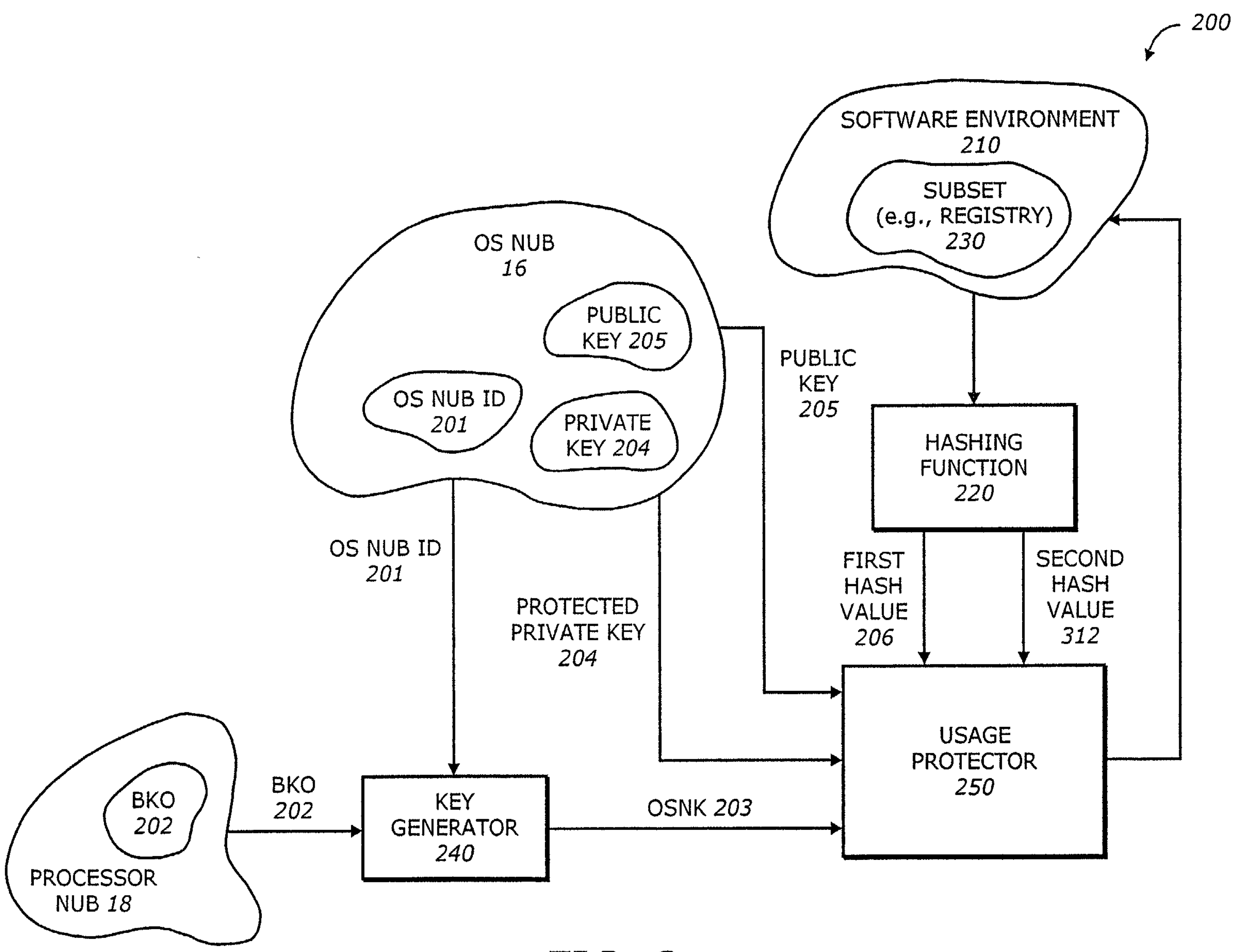

200
SOFTWARE ENVIRONMENT 210
SUBSET (e.g., REGISTRY) 230
OS NUB 16
PUBLIC KEY 205
OS NUB ID 201
PRIVATE KEY 204
PUBLIC KEY 205
HASHING FUNCTION 220
OS NUB ID 201
PROTECTED PRIVATE KEY 204
FIRST HASH VALUE 206
SECOND HASH VALUE 312
USAGE PROTECTOR 250
BKO 202
PROCESSOR NUB 18
BKO 202
KEY GENERATOR 240
OSNK 203

FIG. 2

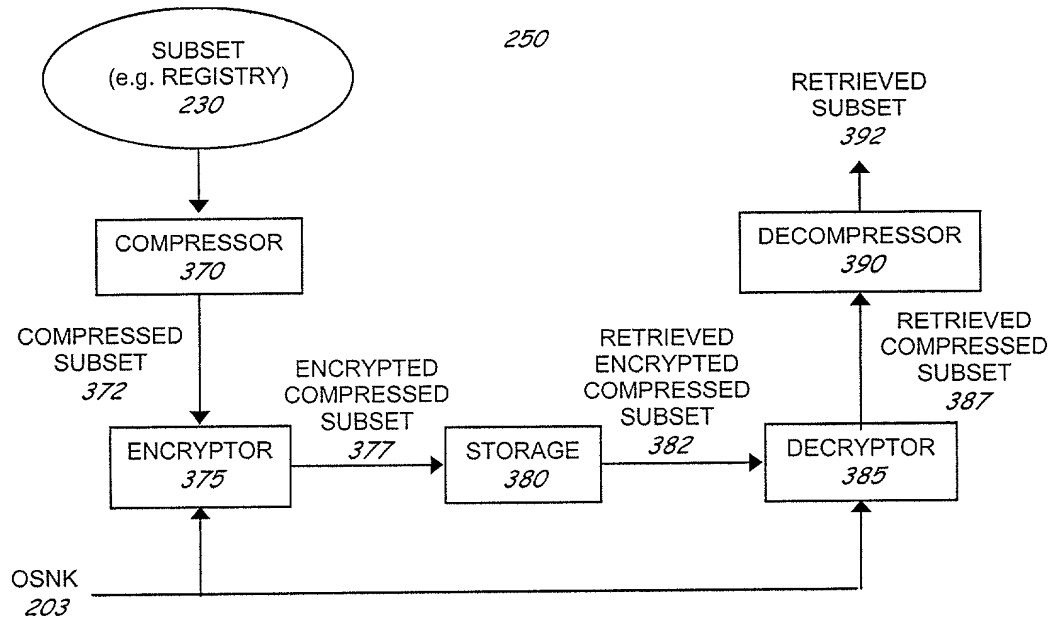

250
SUBSET (e.g. REGISTRY) 230
COMPRESSOR 370
COMPRESSED SUBSET 372
ENCRYPTOR 375
ENCRYPTED COMPRESSED SUBSET 377
STORAGE 380
RETRIEVED ENCRYPTED COMPRESSED SUBSET 382
DECRYPTOR 385
RETRIEVED COMPRESSED SUBSET 387
DECOMPRESSOR 390
RETRIEVED SUBSET 392
OSNK 203

FIGURE 3A

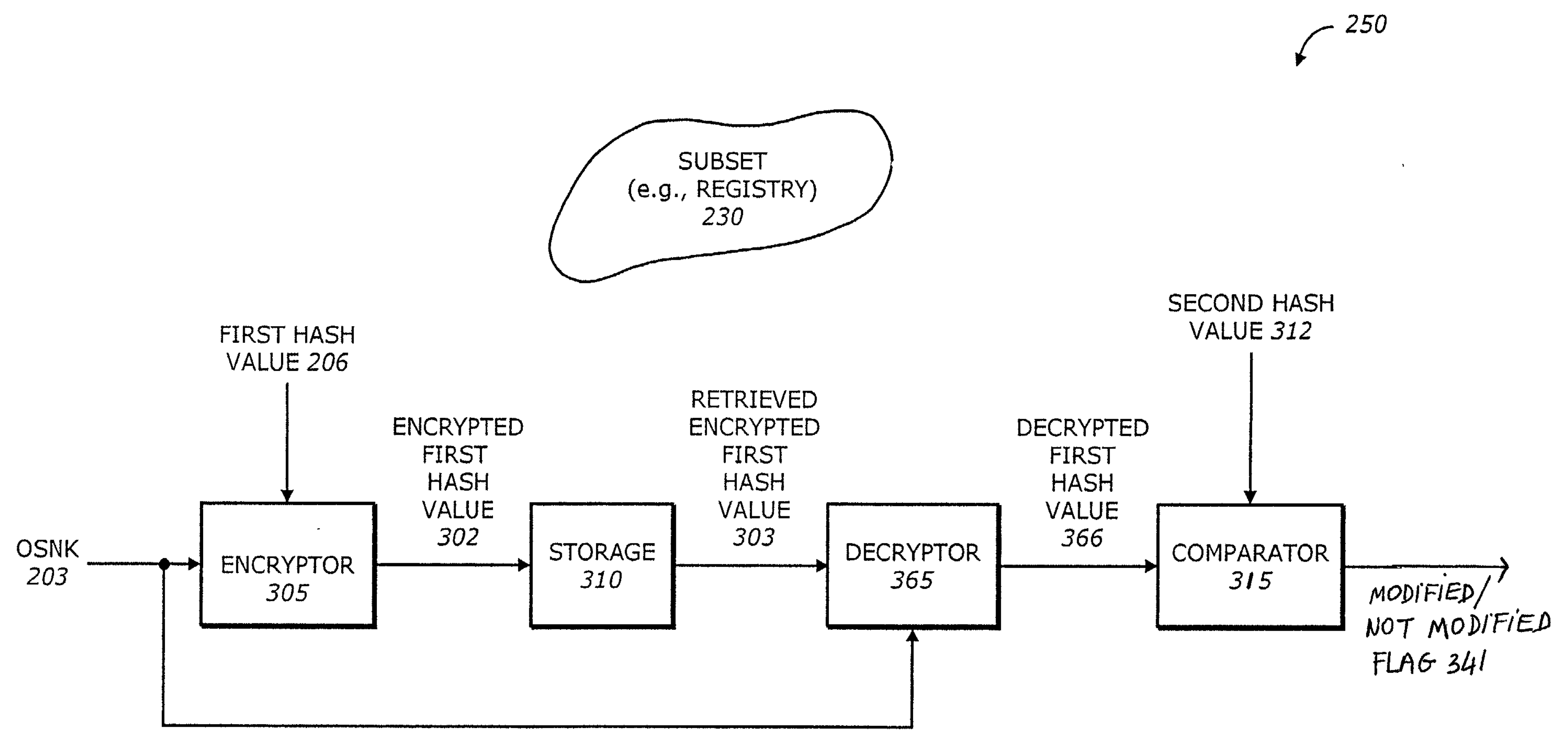

250
SUBSET
(e.g., REGISTRY)
230
FIRST HASH
VALUE 206
OSNK
203
ENCRYPTOR
305
ENCRYPTED
FIRST
HASH
VALUE
302
STORAGE
310
RETRIEVED
ENCRYPTED
FIRST
HASH
VALUE
303
DECRYPTOR
365
DECRYPTED
FIRST
HASH
VALUE
366
SECOND HASH
VALUE 312
COMPARATOR
315
MODIFIED/
NOT MODIFIED
FLAG 341

FIG. 3B

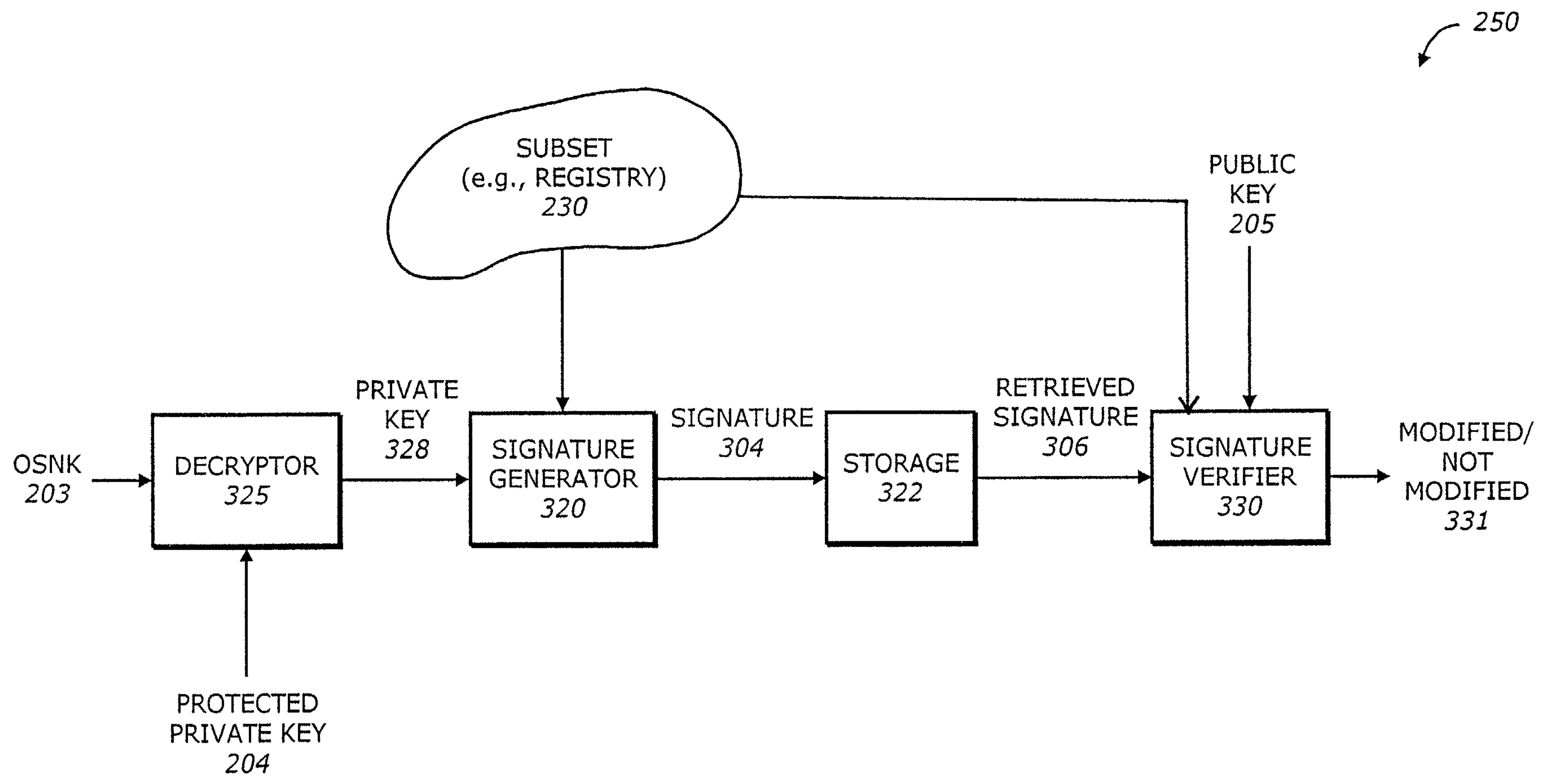

250
SUBSET
(e.g., REGISTRY)
230
PUBLIC
KEY
205
OSNK
203
DECRYPTOR
325
PRIVATE
KEY
328
SIGNATURE
GENERATOR
320
SIGNATURE
304
STORAGE
322
RETRIEVED
SIGNATURE
306
SIGNATURE
VERIFIER
330
MODIFIED/
NOT
MODIFIED
331
PROTECTED
PRIVATE KEY
204

FIG. 3C

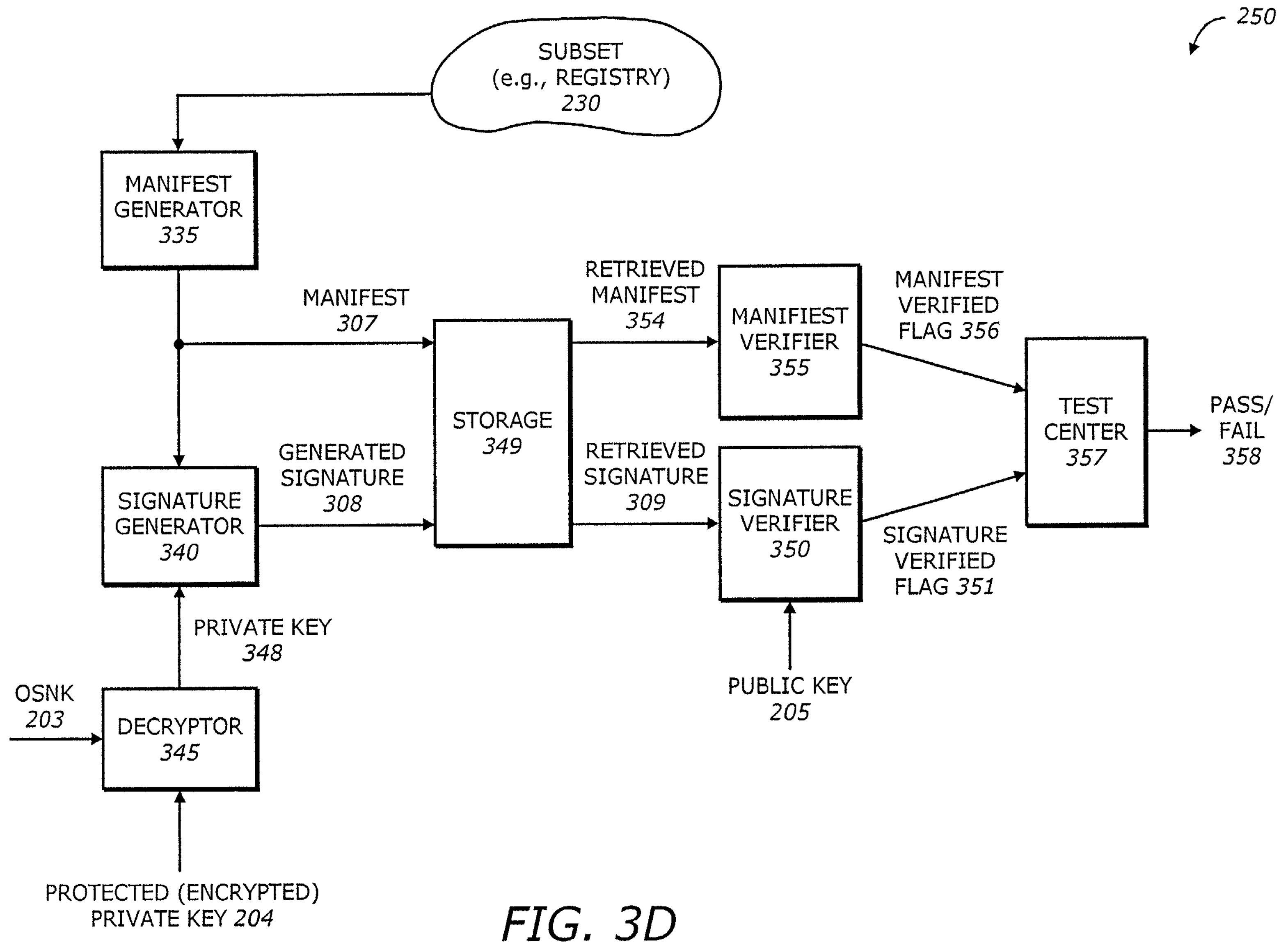

250
SUBSET (e.g., REGISTRY) 230
MANIFEST GENERATOR 335
MANIFEST 307
STORAGE 349
RETRIEVED MANIFEST 354
MANIFIEST VERIFIER 355
MANIFEST VERIFIED FLAG 356
TEST CENTER 357
PASS/ FAIL 358
SIGNATURE GENERATOR 340
GENERATED SIGNATURE 308
RETRIEVED SIGNATURE 309
SIGNATURE VERIFIER 350
SIGNATURE VERIFIED FLAG 351
PRIVATE KEY 348
PUBLIC KEY 205
OSNK 203
DECRYPTOR 345
PROTECTED (ENCRYPTED) PRIVATE KEY 204

FIG. 3D

09668610.092200

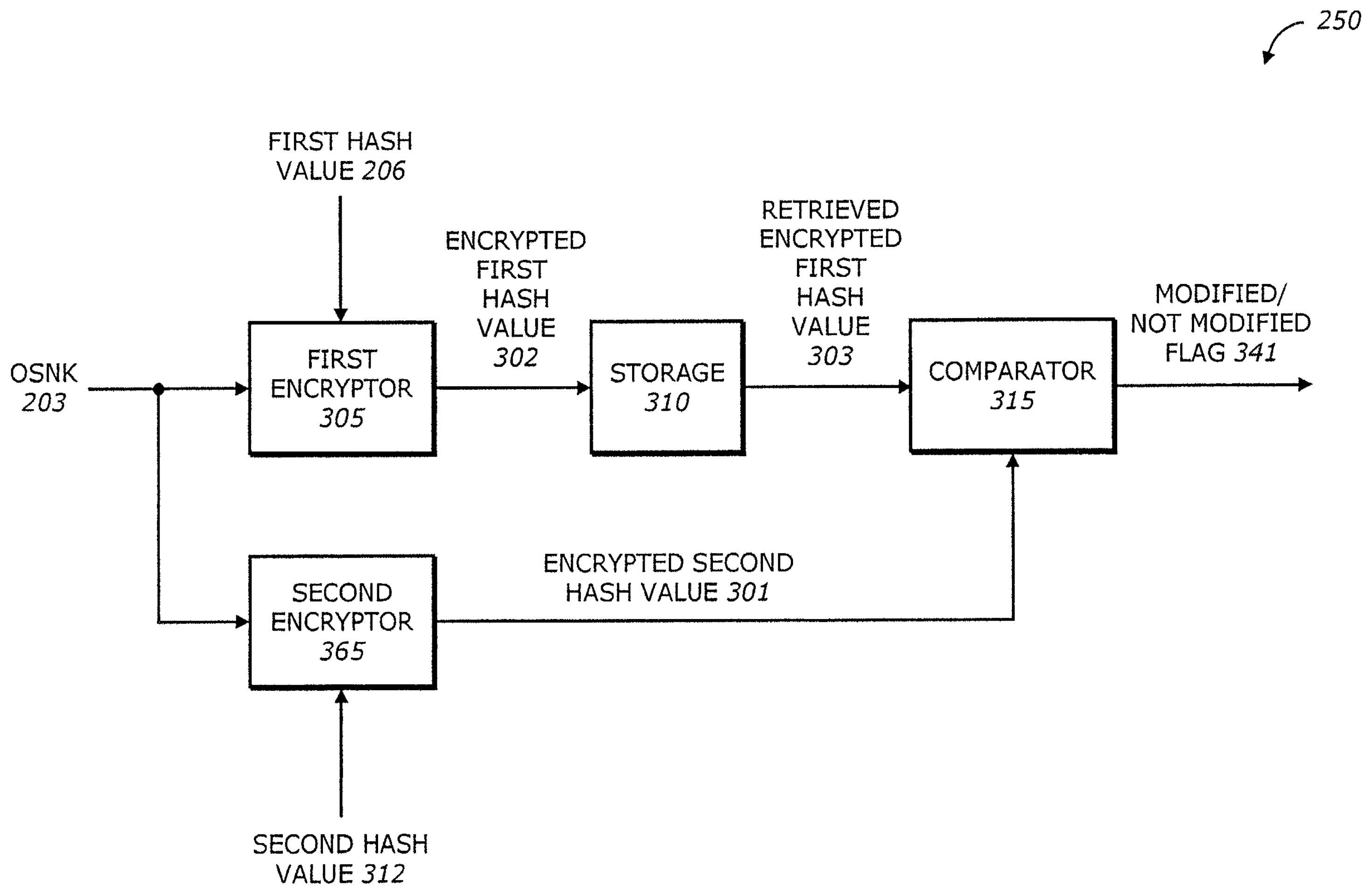
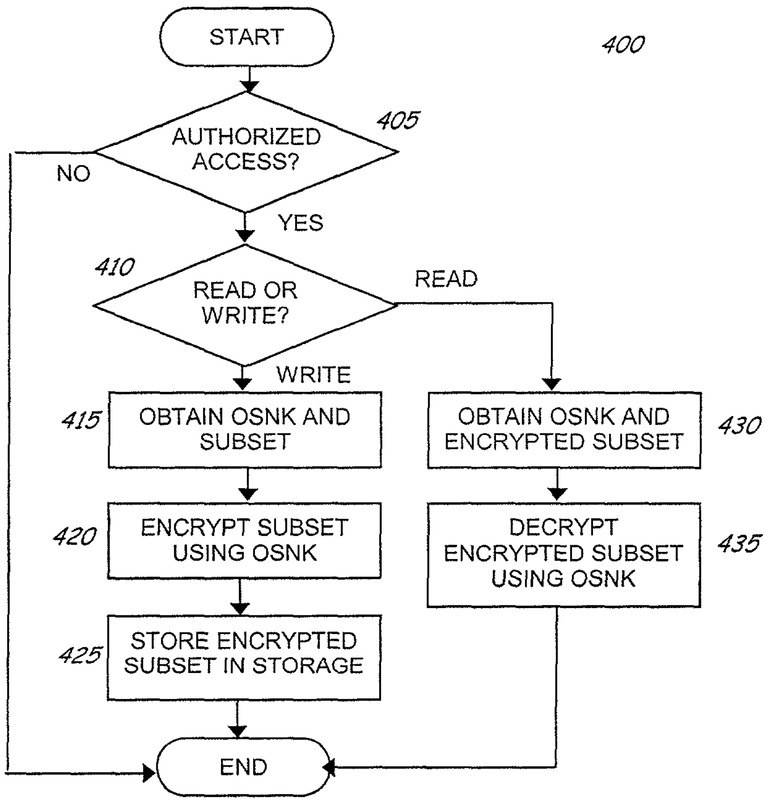

250
FIRST HASH VALUE 206
OSNK 203
FIRST ENCRYPTOR 305
ENCRYPTED FIRST HASH VALUE 302
STORAGE 310
RETRIEVED ENCRYPTED FIRST HASH VALUE 303
COMPARATOR 315
MODIFIED/ NOT MODIFIED FLAG 341
SECOND ENCRYPTOR 365
ENCRYPTED SECOND HASH VALUE 301
SECOND HASH VALUE 312

FIG. 3E



START
400
AUTHORIZED ACCESS?
405
NO
YES
410
READ OR WRITE?
READ
WRITE
415
OBTAIN OSNK AND SUBSET
OBTAIN OSNK AND ENCRYPTED SUBSET
430
420
ENCRYPT SUBSET USING OSNK
DECRYPT ENCRYPTED SUBSET USING OSNK
435
425
STORE ENCRYPTED SUBSET IN STORAGE
END

FIGURE 4

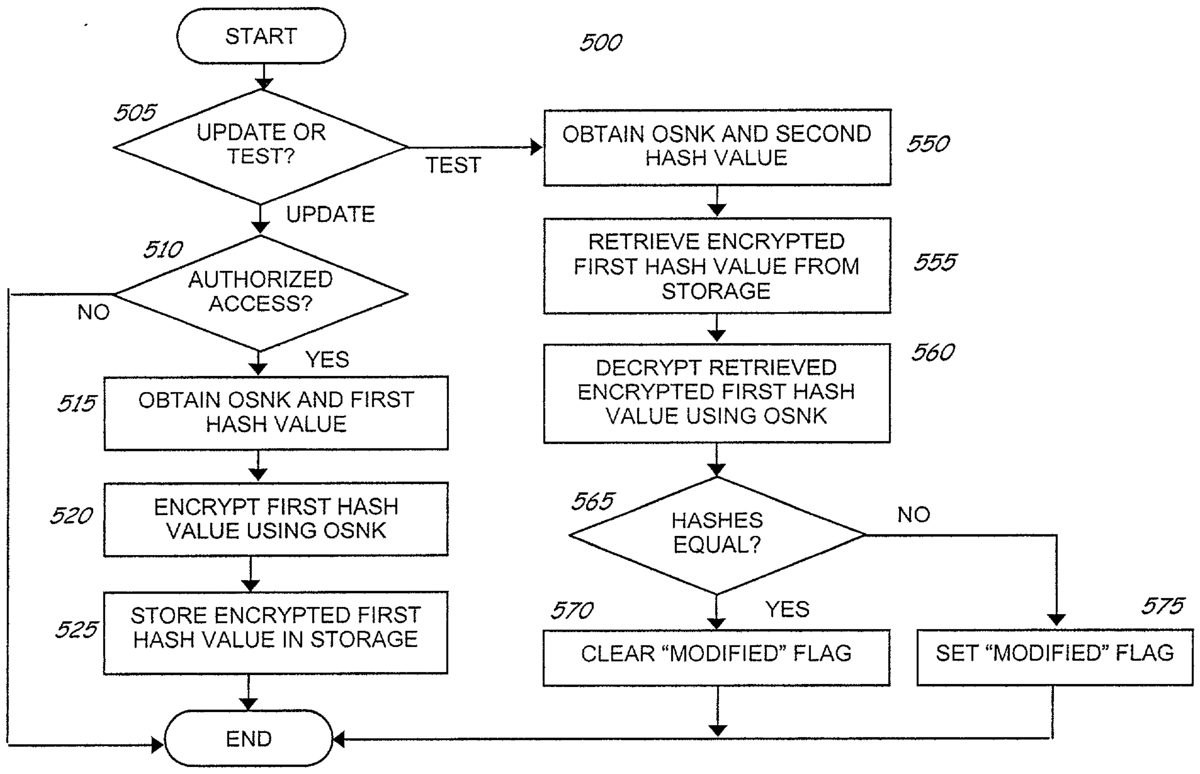

START
500
505
UPDATE OR TEST?
TEST
UPDATE
OBTAIN OSNK AND SECOND HASH VALUE
550
510
AUTHORIZED ACCESS?
NO
YES
RETRIEVE ENCRYPTED FIRST HASH VALUE FROM STORAGE
555
DECRYPT RETRIEVED ENCRYPTED FIRST HASH VALUE USING OSNK
560
515
OBTAIN OSNK AND FIRST HASH VALUE
520
ENCRYPT FIRST HASH VALUE USING OSNK
565
HASHES EQUAL?
NO
525
STORE ENCRYPTED FIRST HASH VALUE IN STORAGE
570
YES
575
CLEAR "MODIFIED" FLAG
SET "MODIFIED" FLAG
END

FIGURE 5

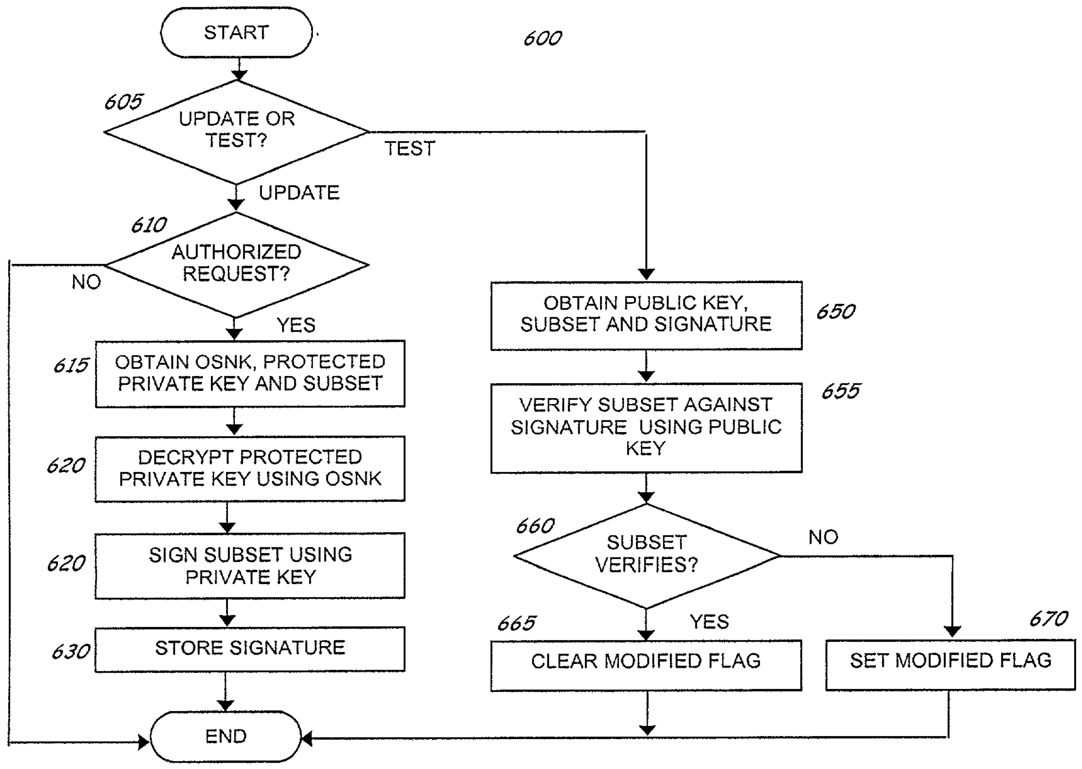
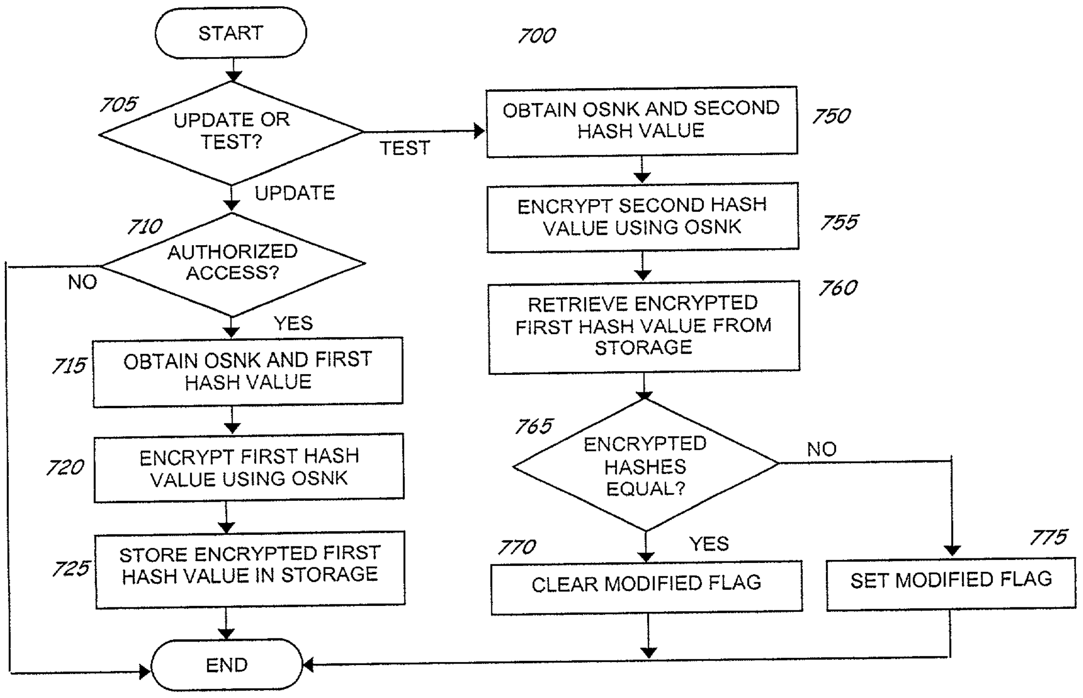

START
600
605
UPDATE OR TEST?
TEST
UPDATE
610
AUTHORIZED REQUEST?
NO
YES
615
OBTAIN OSNK, PROTECTED PRIVATE KEY AND SUBSET
620
DECRYPT PROTECTED PRIVATE KEY USING OSNK
620
SIGN SUBSET USING PRIVATE KEY
630
STORE SIGNATURE
END
OBTAIN PUBLIC KEY, SUBSET AND SIGNATURE
650
VERIFY SUBSET AGAINST SIGNATURE USING PUBLIC KEY
655
660
SUBSET VERIFIES?
NO
YES
665
670
CLEAR MODIFIED FLAG
SET MODIFIED FLAG

FIGURE 6



700
START
705
UPDATE OR TEST?
TEST
UPDATE
710
AUTHORIZED ACCESS?
NO
YES
715
OBTAIN OSNK AND FIRST HASH VALUE
720
ENCRYPT FIRST HASH VALUE USING OSNK
725
STORE ENCRYPTED FIRST HASH VALUE IN STORAGE
END
750
OBTAIN OSNK AND SECOND HASH VALUE
755
ENCRYPT SECOND HASH VALUE USING OSNK
760
RETRIEVE ENCRYPTED FIRST HASH VALUE FROM STORAGE
765
ENCRYPTED HASHES EQUAL?
NO
YES
770
CLEAR MODIFIED FLAG
775
SET MODIFIED FLAG

FIGURE 7

Attorney's Docket No.: 042390.P8104X Patent

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION (CONTINUATION-IN-PART)

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below, next to my name.

I believe I am the original, first, and sole inventor (if only one name is listed below) or any original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

PROTECTING SOFTWARE ENVIRONMENT IN ISOLATED EXECUTION

the specification of which

[X] is attached hereto.
[ ] was filed on ______________ as
United States Application Number ______________
or PCT International Application Number ______________
and was amended on ______________
(if applicable)

09539510.052200

That this application in part discloses and claims subject matter disclosed in my earlier filed pending application:

Application No.: 09/540,946
Filed: 3/31/2000

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

That as to the subject matter of this application which is common to said earlier application, I do not know and do not believe that the same was ever known or used in the United States of America before my invention thereof, or patented or described in any printed publication in any country before my invention thereof or more than one year prior to this application, that the same was not in public use or on sale in the United States of America more than one year prior to this application, and

That said common subject matter has not been patented or made the subject of an inventor's certificate issued before the date of said earlier application in any country foreign to the United States of America on an application filed by me or my legal representatives or assigns more than twelve months prior to said earlier application;

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119, of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

APPLICATION NUMBER	COUNTRY (OR INDICATE IF PCT)	DATE OF FILING (day, **month**, year)	PRIORITY CLAIMED UNDER 37 USC 119
			☐ No ☐ Yes
			☐ No ☐ Yes
			☐ No ☐ Yes

<u>I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below</u> and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

APPLICATION NUMBER	FILING DATE	STATUS (ISSUED, PENDING, ABANDONED)

I hereby appoint the persons listed on Appendix A hereto (which is incorporated by reference and a part of this document) as my respective patent attorneys and patent agents, with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

Send correspondence to:
Thinh V. Nguyen, Reg. No. 42,034, BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, LLP
(Name of Attorney or Agent)
12400 Wilshire Boulevard, 7th Floor, Los Angeles, California 90025 and direct telephone calls to:
<u>Thinh V. Nguyen</u>, (714) 557-3800.
(Name of Attorney or Agent)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor (given name, family name) **Carl M. Ellison**

Inventor's Signature Carl M. Ellison Date 8/18/00

Residence Portland, Oregon USA *(City, State)* Citizenship USA *(Country)*

P. O. Address 1818 N.W. 28th Avenue
Portland, Oregon 97210-2214 USA

Full Name of Second/Joint Inventor (given name, family name) **Roger A. Golliver**

Inventor's Signature ______ Date ______

Residence Beaverton, Oregon USA *(City, State)* Citizenship USA *(Country)*

P. O. Address 16185 SW Night Hawk Drive
Beaverton, Oregon 97007-8368 USA

<u>I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below</u> and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

APPLICATION NUMBER	FILING DATE	STATUS (ISSUED, PENDING, ABANDONED)

I hereby appoint the persons listed on Appendix A hereto (which is incorporated by reference and a part of this document) as my respective patent attorneys and patent agents, with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

Send correspondence to:
Thinh V. Nguyen, Reg. No. 42,034, BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, LLP
(Name of Attorney or Agent)
12400 Wilshire Boulevard, 7th Floor, Los Angeles, California 90025 and direct telephone calls to:
<u>Thinh V. Nguyen</u>, (714) 557-3800.
(Name of Attorney or Agent)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor (given name, family name) **Carl M. Ellison**

Inventor's Signature ______________________ Date ____________

Residence Portland, Oregon USA *(City, State)* Citizenship USA *(Country)*

P. O. Address 1818 N.W. 28th Avenue
Portland, Oregon 97210-2214 USA

Full Name of Second/Joint Inventor (given name, family name) **Roger A. Golliver**

Inventor's Signature [illegible] Date 7/28/2006

Residence Beaverton, Oregon USA *(City, State)* Citizenship USA *(Country)*

P. O. Address 16185 SW Night Hawk Drive
Beaverton, Oregon 97007-8368 USA

Full Name of Third/Joint Inventor (given name, family name) **Howard C. Herbert**

Inventor's Signature Howard C. Herbert Date 24 July 2000

Residence Phoenix, Arizona USA Citizenship USA
(City, State) (Country)

P. O. Address 16817 South 1st Drive
Phoenix, Arizona 85045 USA

Full Name of Fourth/Joint Inventor (given name, family name) **Derrick C. Lin**

Inventor's Signature ______________ Date ______________

Residence Foster City, California USA Citizenship USA
(City, State) (Country)

P. O. Address 113 Barkentine Street
Foster City, California 94404 USA

Full Name of Fifth/Joint Inventor (given name, family name) **Francis X. McKeen**

Inventor's Signature ______________ Date ______________

Residence Portland, Oregon USA Citizenship USA
(City, State) (Country)

P. O. Address 10612 N. W. LeMans Ct.
Portland, Oregon 97229 USA

Full Name of Sixth/Joint Inventor (given name, family name) **Gilbert N. Neiger**

Inventor's Signature ______________ Date ______________

Residence Portland, Oregon USA Citizenship USA
(City, State) (Country)

P. O. Address 2424 N. E. 11th Avenue
Portland, Oregon 97212 USA

Full Name of Third/Joint Inventor (given name, family name)	**Howard C. Herbert**		
Inventor's Signature		Date	
Residence	Phoenix, Arizona USA *(City , State)*	Citizenship	USA *(Country)*
P. O. Address	16817 South 1st Drive Phoenix, Arizona 85045 USA		

Full Name of Fourth/Joint Inventor (given name, family name)	**Derrick C. Lin**		
Inventor's Signature	[illegible]	Date	8/30/00
Residence	San Mateo, California USA *(City , State)*	Citizenship	USA *(Country)*
P. O. Address	1737 Oakwood Drive San Mateo, California 94403 USA		

Full Name of Fifth/Joint Inventor (given name, family name)	**Francis X. McKeen**		
Inventor's Signature		Date	
Residence	Portland, Oregon USA *(City , State)*	Citizenship	USA *(Country)*
P. O. Address	10612 N. W. LeMans Ct. Portland, Oregon 97229 USA		

Full Name of Sixth/Joint Inventor (given name, family name)	**Gilbert Neiger**		
Inventor's Signature		Date	
Residence	Portland, Oregon USA *(City , State)*	Citizenship	USA *(Country)*
P. O. Address	2424 N. E. 11th Avenue Portland, Oregon 97212 USA		

Full Name of Third/Joint Inventor (given name, family name) **Howard C. Herbert**

Inventor's Signature ______________________ Date ____________

Residence Phoenix, Arizona USA *(City, State)* Citizenship USA *(Country)*

P. O. Address 16817 South 1st Drive
Phoenix, Arizona 85045 USA

Full Name of Fourth/Joint Inventor (given name, family name) **Derrick C. Lin**

Inventor's Signature ______________________ Date ____________

Residence Foster City, California USA *(City, State)* Citizenship USA *(Country)*

P. O. Address 113 Barkentine Street
Foster City, California 94404 USA

Full Name of Fifth/Joint Inventor (given name, family name) **Francis X. McKeen**

Inventor's Signature Francis X. McKeen Date 8/3/00

Residence Portland, Oregon USA *(City, State)* Citizenship USA *(Country)*

P. O. Address 10612 N. W. LeMans Ct.
Portland, Oregon 97229 USA

Full Name of Sixth/Joint Inventor (given name, family name) **Gilbert N. Neiger**

Inventor's Signature ______________________ Date ____________

Residence Portland, Oregon USA *(City, State)* Citizenship USA *(Country)*

P. O. Address 2424 N. E. 11th Avenue
Portland, Oregon 97212 USA

Full Name of Third/Joint Inventor (given name, family name)	**Howard C. Herbert**		
Inventor's Signature		Date	
Residence	Phoenix, Arizona USA *(City, State)*	Citizenship	USA *(Country)*
P. O. Address	16817 South 1st Drive Phoenix, Arizona 85045 USA		

Full Name of Fourth/Joint Inventor (given name, family name)	**Derrick C. Lin**		
Inventor's Signature		Date	
Residence	Foster City, California USA *(City, State)*	Citizenship	USA *(Country)*
P. O. Address	113 Barkentine Street Foster City, California 94404 USA		

Full Name of Fifth/Joint Inventor (given name, family name)	**Francis X. McKeen**		
Inventor's Signature		Date	
Residence	Portland, Oregon USA *(City, State)*	Citizenship	USA *(Country)*
P. O. Address	10612 N. W. LeMans Ct. Portland, Oregon 97229 USA		

Full Name of Sixth/Joint Inventor (given name, family name)	**Gilbert Neiger**		
Inventor's Signature	[illegible]	Date	7/27/2000
Residence	Portland, Oregon USA *(City, State)*	Citizenship	USA *(Country)*
P. O. Address	2424 N. E. 11th Avenue Portland, Oregon 97212 USA		

Docket No.

Full Name of Seventh/Joint Inventor (given name, family name)	**Ken Reneris**
Inventor's Signature [illegible]	Date 8/3/2000
Residence Wilbraham, Massachusetts USA *(City , State)*	Citizenship USA *(Country)*
P. O. Address 8 Red Gap Road, Wilbraham, Massachusetts 01095 USA	

Full Name of Eighth/Joint Inventor (given name, family name)	**James A. Sutton**
Inventor's Signature	Date
Residence Portland, Oregon USA *(City , State)*	Citizenship USA *(Country)*
P. O. Address 20205 N. W. Paulina Drive, Portland, Oregon 97229 USA	

Full Name of Ninth/Joint Inventor (given name, family name)	**Shreekant S. Thakkar**
Inventor's Signature	Date
Residence Portland, Oregon USA *(City , State)*	Citizenship United Kingdom *(Country)*
P. O. Address 150 S.W. Moonridge Place, Portland, Oregon 92775 USA	

Full Name of Tenth/Joint Inventor (given name, family name)	**Millind Mittal**
Inventor's Signature	Date
Residence Palo Alto, CA USA *(City , State)*	Citizenship USA *(Country)*
P. O. Address 800 E. Charleston Road, #29, Palo Alto, CA 94303 USA	

Full Name of Seventh/Joint Inventor (given name, family name)	**Ken Reneris**		
Inventor's Signature		Date	
Residence	Wilbraham, Massachusetts USA *(City , State)*	Citizenship	USA *(Country)*
P. O. Address	8 Red Gap Road		
	Wilbraham, Massachusetts 01095 USA		

Full Name of Eighth/Joint Inventor (given name, family name)	**James A. Sutton**		
Inventor's Signature	~~7-24-2000~~ [illegible]	Date	7-24-2000
Residence	Portland, Oregon USA *(City , State)*	Citizenship	USA *(Country)*
P. O. Address	20205 N. W. Paulina Drive		
	Portland, Oregon 97229 USA		

Full Name of Ninth/Joint Inventor (given name, family name)	**Shreekant S. Thakkar**		
Inventor's Signature		Date	
Residence	Portland, Oregon USA *(City , State)*	Citizenship	United Kingdom *(Country)*
P. O. Address	150 S.W. Moonridge Place		
	Portland, Oregon 92775 USA		

Full Name of Tenth/Joint Inventor (given name, family name)	**Millind Mittal**		
Inventor's Signature		Date	
Residence	Palo Alto, CA USA *(City , State)*	Citizenship	USA *(Country)*
P. O. Address	800 E. Charleston Road, #29		
	Palo Alto, CA 94303 USA		

Full Name of Seventh/Joint Inventor (given name, family name) **Ken Reneris**

Inventor's Signature ______________ Date ______________

Residence Wilbraham, Massachusetts USA
(City , State)

Citizenship USA
(Country)

P. O. Address 8 Red Gap Road
Wilbraham, Massachusetts 01095 USA

Full Name of Eighth/Joint Inventor (given name, family name) **James A. Sutton**

Inventor's Signature ______________ Date ______________

Residence Portland, Oregon USA
(City , State)

Citizenship USA
(Country)

P. O. Address 20205 N. W. Paulina Drive
Portland, Oregon 97229 USA

Full Name of Ninth/Joint Inventor (given name, family name) **Shreekant S. Thakkar**

Inventor's Signature [illegible] Date ______________

Residence Portland, Oregon USA
(City , State)

Citizenship United Kingdom
(Country)

P. O. Address 150 S.W. Moonridge Place
Portland, Oregon 92775 USA

Full Name of Tenth/Joint Inventor (given name, family name) **Millind Mittal**

Inventor's Signature ______________ Date ______________

Residence Palo Alto, CA USA
(City , State)

Citizenship USA
(Country)

P. O. Address 800 E. Charleston Road, #29
Palo Alto, CA 94303 USA

Full Name of Tenth/Joint Inventor (given name, family name)			**Millind Mittal**
Inventor's Signature	[illegible]	Date	9/5/00
Residence	Palo Alto, CA USA (City, State)	Citizenship	USA (Country)
P. O. Address	800 E. Charleston Road, #29 Palo Alto, CA 94303 USA		

Full Name of Eleventh/Joint Inventor (given name, family name)			
Inventor's Signature		Date	
Residence	(City, State)	Citizenship	(Country)
P. O. Address			

09653410.090500

Appendix A

I hereby appoint BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP, a firm including: William E. Alford, Reg. No. 37,764; Farzad E. Amini, Reg. No. 42,261; William Thomas Babbitt, Reg. No. 39,591; Carol F. Barry, Reg. No. 41,600; Jordan Michael Becker, Reg. No. 39,602; Lisa N. Benado, Reg. No. 39,995; Bradley J. Bereznak, Reg. No. 33,474; Michael A. Bernadicou, Reg. No. 35,934; Roger W. Blakely, Jr., Reg. No. 25,831; R. Alan Burnett, Reg. No. 46,149; Gregory D. Caldwell, Reg. No. 39,926; Andrew C. Chen, Reg. No. 43,544; Thomas M. Coester, Reg. No. 39,637; Donna Jo Coningsby, Reg. No. 41,684; Dennis M. deGuzman, Reg. No. 41,702; Stephen M. De Klerk, Reg. No. P46,503; Michael Anthony DeSanctis, Reg. No. 39,957; Daniel M. De Vos, Reg. No. 37,813; Sanjeet Dutta, Reg. No. P46,145; Matthew C. Fagan, Reg. No. 37,542; Tarek N. Fahmi, Reg. No. 41,402; George Fountain, Reg. No. 36,374; Paramita Ghosh, Reg. No. 42,806; James Y. Go, Reg. No. 40,621; James A. Henry, Reg. No. 41,064; Willmore F. Holbrow III, Reg. No. P41,845; Sheryl Sue Holloway, Reg. No. 37,850; George W Hoover II, Reg. No. 32,992; Eric S. Hyman, Reg. No. 30,139; William W. Kidd, Reg. No. 31,772; Sang Hui Kim, Reg. No. 40,450; Walter T. Kim, Reg. No. 42,731; Eric T. King, Reg. No. 44,188; Erica W. Kuo, Reg. No. 42,775; George B. Leavell, Reg. No. 45,436; Gordon R. Lindeen III, Reg. No. 33,192; Jan Carol Little, Reg. No. 41,181; Kurt P. Leyendecker, Reg. No. 42,799; Joseph Lutz, Reg. No. 43,765; Michael J. Mallie, Reg. No. 36,591; Andre L. Marais, under 37 C.F.R. § 10.9(b); Paul A. Mendonsa, Reg. No. 42,879; Clive D. Menezes, Reg. No. 45,493; Chun M. Ng, Reg. No. 36,878; Thien T. Nguyen, Reg. No. 43,835; Thinh V. Nguyen, Reg. No. 42,034; Dennis A. Nicholls, Reg. No. 42,036; Daniel E. Ovanezian, Reg. No. 41,236; Kenneth B. Paley, Reg. No. 38,989; Marina Portnova, Reg. No. P45,750; William F. Ryann, Reg. 44,313; James H. Salter, Reg. No. 35,668; William W. Schaal, Reg. No. 39,018; James C. Scheller, Reg. No. 31,195; Jeffrey Sam Smith, Reg. No. 39,377; Maria McCormack Sobrino, Reg. No. 31,639; Stanley W. Sokoloff, Reg. No. 25,128; Judith A. Szepesi, Reg. No. 39,393; Vincent P. Tassinari, Reg. No. 42,179; Edwin H. Taylor, Reg. No. 25,129; John F. Travis, Reg. No. 43,203; Joseph A. Twarowski, Reg. No. 42,191; Thomas A. Van Zandt, Reg. No. 43,219; Lester J. Vincent, Reg. No. 31,460; Glenn E. Von Tersch, Reg. No. 41,364; John Patrick Ward, Reg. No. 40,216; Mark L. Watson, Reg. No. P46,322; Thomas C. Webster, Reg. No. P46,154; and Norman Zafman, Reg. No. 26,250; my patent attorneys, and Firasat Ali, Reg. No. 45,715; and Justin M. Dillon, Reg. No. 42,486; my patent agents, with offices located at 12400 Wilshire Boulevard, 7th Floor, Los Angeles, California 90025, telephone (714) 557-3800, with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.